January 2019

Design Of A Controlled Descent Lifting Body Glider For High Altitude Payload Recovery

Nanette Valentour

DESIGN OF A CONTROLLED DESCENT LIFTING BODY GLIDER FOR HIGH ALTITUDE PAYLOAD RECOVERY

by

Nanette Bassett Valentour
Bachelor of Science, University of Cincinnati, 2016

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota

December
2019

This thesis, submitted by Nanette Valentour in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

Dr. Pablo de Leon

Dr. James Casler

Dr. Forrest Ames

This thesis is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

Chris Nelson
Associate Dean of the School of Graduate Studies

12/2/19
Date

PERMISSION

Title	Design of a Controlled Descent Lifting Body Glider for High Altitude Payload Recovery
Department	Space Studies
Degree	Master of Science

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in his absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Nanette Valentour
November 9, 2018

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my advisor and members of my advisory Committee for their guidance and support during my time in the master's program at the University of North Dakota.

ABSTRACT

A lifting body scaled glider to be used as a recovery method for high altitude ballooning payloads has been developed at the University of North Dakota. Current recovery techniques for balloon payloads consist of monitoring and chasing the payload signal until it can be recovered. It is expected that a lifting body glider can reduce the distance for payload recovery and in some cases effectively return the payload back to the Ground Station. A preliminary subsonic aerodynamic model has been developed using MATLAB and the Vortex Lattice Method (VLM). Traditional evaluation techniques typical for calculation of the aerodynamic coefficients are not sufficient to predict the flight performance of the glider due to its fuselage lift contribution. Thus, two-dimensional panel methods along with the traditional three-dimensional VLM are used in conjunction to produce adequate results. The proposed computational model is expected to sufficiently estimate the subsonic aerodynamic characteristics of the glider to allow for the optimization of the geometry. The calculated loads will be used to design the structure of the glider, which is fabricated using composites and the appropriate resin. The glider will be integrated with a separately developed navigation system before undergoing a preliminary low altitude flight test for verification of the computational model and structural design. The aerodynamic coefficients will be compiled into functions which will be used in the MATLAB Aerospace Module and Simulink Blockset to create a robust model for simulation studies. The model must be developed such that the operator can simulate different flight paths and optimize the carrying capacity of the glider to support various payloads used for atmospheric experimentation. The resulting simulation will allow the user to analyze and trim the glider for various flight configurations depending on the location of the center of mass and the anticipated release altitude. For high altitude ballooning students unfamiliar with computational

aerodynamics, the simulation model will help the student to visualize how difficult or easy the glider would be to control under the current configuration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

NOMENCLATURE

C_{D_0}	Zero-lift drag coefficient
C_{D_a}	Change in drag coefficient due to the rate of change of the angle of attack
C_{D_u}	Change in drag coefficient due to forward speed
C_{L_0}	Reference lift coefficient
C_{L_u}	Change in lift coefficient due to forward speed
C_{m_a}	Change in pitching moment coefficient due to the rate of change of the angle of attack
C_A	Axial-force coefficient
C_D	Drag coefficient
C_L	Lift coefficient
C_N	Normal-force coefficient
C_X	Axial-force coefficient
C_Y	Side-force coefficient
C_Z	Normal-force coefficient
C_l	Rolling-moment coefficient
C_m	Pitching-moment coefficient
C_n	Yawing-moment coefficient
C_p	Center of pressure
F_X	Aerodynamic force in x-direction
F_Y	Aerodynamic force in y-direction
F_Z	Aerodynamic force in z-direction
I_x	Mass moment of inertia about x-axis

I_y	Mass moment of inertia about y-axis
I_z	Mass moment of inertia about z-axis
$\frac{L}{D}$	Lift over drag
L_p	Dimensional derivative of the change in rolling moment due to roll rate
L_r	Dimensional derivative of the change in rolling moment due to yaw rate
L_v	Dimensional derivative of the change in rolling moment due to lateral speed
L_β	Dimensional derivative of the change in rolling moment due to change of sideslip angle
M_X	Moment about x-axis
M_Y	Moment about y-axis
M_Z	Moment about z-axis
M_q	Dimensional derivative of the change in pitching moment due to pitching velocity
M_u	Dimensional derivative of the change in pitching moment due to forward speed
M_w	Dimensional derivative of the change in pitching moment due to vertical speed
N_p	Dimensional derivative of the change in yawing moment due to roll rate
N_r	Dimensional derivative of the change in yawing moment due to yaw rate
N_v	Dimensional derivative of the change in yawing moment due to lateral speed
N_β	Dimensional derivative of the change in yawing moment due to change of sideslip angle
S_{ref}	Reference area
X_u	Dimensional derivative of the change in x-force due to forward speed
X_w	Dimensional derivative of the change in x-force due to lateral speed
Y_p	Dimensional derivative of the change in y-force due to roll rate
Y_r	Dimensional derivative of the change in y-force due to yaw rate

Y_v	Dimensional derivative of the change in y-force due to lateral speed
Y_β	Dimensional derivative of the change in y-force due to the change of sideslip angle
Z_u	Dimensional derivative of the change in z-force due to forward speed
Z_w	Dimensional derivative of the change in z-force due to vertical speed
b_{ref}	Reference span
$c_{mac}, \bar{c}$	Mean aerodynamic chord
d_{ref}	Reference length
$\bar{q}$	Dynamic pressure
δ_a	Aileron control surface deflection
δ_{bf}	Body flap control surface deflection
δ_e	Elevator control surface deflection
θ_{min}	Minimum glide angle
σ_Y	Material yield stress
h	Height of center of gravity above ground
Γ	Vortex strength
AR	Aspect ratio, b^2/S
B	Boom area
$F.S.$	Factor of safety
M	Mach number
P	Curve-fit polynomial coefficient vector
S	Section modulus
b	Span

c	Chord
p	Roll rate
q	Pitch rate
r	Yaw rate
x	x-coordinate
y	y-coordinate
z	z-coordinate
$\boldsymbol{I}$	Identity matrix
α	Angle of attack
β	Angle of sideslip
θ	Pitch angle
λ	Taper ratio, c_{tip}/c_{root}
ρ	Density
σ	Material stress
τ	Shear stress
φ	Roll angle
ψ	Yaw angle

CHAPTER 1. INTRODUCTION

1.1 Statement of the Problem

High altitude ballooning is a popular method to deploy atmospheric experiments to the edges of the atmosphere. Current recovery techniques consist of monitoring and chasing the payload signal until it can be recovered. Some payloads can drift for hundreds of miles before landing, requiring a large amount of time and effort to recover the ballooning payload. Current mitigation techniques consist of limiting the time of launch to a window where the weather conditions are satisfactory to recover the payload in a somewhat timely manner. However, the imposed limitations reduce the amount of experiments that can be completed.

1.2 Research Objective

A scaled lifting body glider was chosen as a proposed aerodynamically suitable design for the recovery of high altitude atmospheric payloads. A controlled descent vehicle is desirable for this application because it would support a larger number of experiments in the allotted timeframe and create a platform for new atmospheric testing. However, the lifting body vehicle is not a typical aircraft. Conventional aircraft generate lift from the wings, while lifting bodies generate lift from their flattened fuselages. This makes theoretical aerodynamic analysis more difficult because traditional evaluation techniques may not be applicable. Thus, a computational analysis of the aerodynamic characteristics of the glider was conducted to determine the proposed vehicle's stability and efficiency. The purpose of this research is to demonstrate whether a lifting body scaled model is a suitable aerodynamic design that can be used as a recovery method for high altitude ballooning payloads.

CHAPTER 2. A REVIEW OF THE LITERATURE

2.1 Previous Research

DeLeon [1] conducted a high altitude drop test using a ballistic capsule and high altitude balloons. The objectives of this research were to understand the development of a reusable sub-orbital vehicle, learn about instrumentation, guidance, and recovery procedures, and to test experiments in a reduced gravity environment. At the completion of this research, de Leon suggested a lifting body configuration to replace the ballistic capsule. The instrumentation and a video feed could then be used to glide the vehicle back to the landing area.

A subsonic aerodynamic model of the HL-20 lifting body was compiled by Jackson [2] to develop a flight simulation using MATLAB and Simulink. Data from wind tunnel testing were compiled into polynomial functions which serve as look up tables in Simulink. The data were trimmed and linearized in MATLAB/Simulink to create an autopilot glide-to-landing flight simulation [3].

2.2 Lifting Bodies

The lifting body, a fixed wing aircraft for which the body produces lift, was a major area of research in the 60's and 70's as a way to build small lightweight manned spacecraft. These spacecraft were designated as a personnel launch system to be complementary to the space shuttle. In the event the shuttle was not available, manned access to space would remain available. It was believed that the smaller compact vehicle designed specifically for the crew only would increase crew safety, as compared to the larger and more complex space shuttle which included main propulsion engines and a payload bay. Compared to current ballistic re-entry techniques, lifting bodies offer the flexibility of a controlled descent and the ability to land on any runway.

2.3 Vortex Panel Methods

Lift on a wing is produced due to the pressure differences over the upper and lower surfaces. This pressure distribution results in the air on the upper surface flowing inboard toward the root, while the air on the lower surface flows outward towards the wing tips. Thus, there is a flow around the finite wing having both chordwise and spanwise velocity components. As the flows from the upper and lower surfaces join at the trailing edge of the wing, the difference in spanwise velocity components will cause the air to roll up into a number of streamwise vortices distributed along the span [4]. The small streamwise vortices roll up into two large vortices at the wingtips and are visible if condensation is present during flight, shown in Figure 1.

Figure 1. Wing-tip vortices of the Space Shuttle Orbiter Columbia [4].

The lift force along the span of the wing is related to the strength of the spanwise circulation distribution, resulting in both the circulation and lift reducing to zero at the wing tip. Mathematical procedures can be used to determine the vortex-strength distribution produced by the flow field of a wing, which can be used for the computation of the aerodynamic forces and coefficients.

The traditional Vortex Lattice Method (VLM) represents the wing by overlaying a grid of horseshoe vortices on the mean surface. Each horseshoe vortex induces a velocity at the control point at which the velocities can be calculated using the Biot-Savart Law. A boundary condition is then applied to the wing such that no flow can go through the wing. Subsequent summation of the velocities at the control points produces a set of mathematical equations for the horseshoe strengths with respect to the boundary condition. The calculated vortex strengths, which are related to the pressure differential over the upper and lower surfaces, are integrated to produce the total forces and moments.

The force acting on a surface in fluid flow is represented by two components. The component perpendicular to the surface is the pressure force and the parallel component is the shear, or friction, force. The pressure forces acting along the normal surface are composed of the static and dynamic pressures. When the surface is at rest, the static and total pressures are equal at every point on the surface. When the surface is in motion, the surface creates a flow field in the air because the fluid moves around the impenetrable surface. The flow field is highly dependent on the geometry of the surface, with the static pressure higher for areas facing the wind than for areas parallel to the wind [5]. This phenomenon accounts for the pressure forces on the surface.

The viscous motion of the boundary layer over the surface is the source of the shear forces. This is because the velocity of the air closest to the surface is equal to zero, but the velocity increases along the normal until the air flow is equivalent to the free stream velocity. This microscopic change in velocity creates the boundary layer, which is often very small with respect to the dimensions of the lifting surface. The two forces are usually considered separately and then combined to form the aerodynamic properties of the surface. The following analysis

only considered the pressure forces because they are dominant in the linear domain [5]. Linear aerodynamics is the field of aerodynamics which only consists of linear behavior. There are limitations, but during takeoff and landing all aircraft will behave linearly. It is useful for this analysis because a linear domain assumes low speeds below the stall angle, neglecting compressible flow effects.

A vortex flow is a potential flow where all the streamlines are concentric circles about a given point where the velocity at any point along the streamline is constant. The velocities will vary from one streamline to another with respect to the distance from a common center. The vortex flow is a physically possible incompressible flow, where no mass is produced in the field, i.e., $\nabla \cdot \vec{V} = 0$ at every point, and irrotational, i.e., $\nabla \times \vec{V} = 0$, at every point except the origin [6]. At the origin exists a singularity in the field, which is interpreted to be a point vortex which induces about it the circular vortex flow. Imagine a line going straight through the page, at the singularity point, extending to infinity on either side. This line is a straight vortex filament of constant strength Γ.

CHAPTER 3. COMPUTATIONAL AERODYNAMIC MODELING

3.1 Preliminary Aerodynamic Model Development

The computational procedure is built up using MATLAB and accessible examples and data from existing aircraft. At each step of the process, the developed code is verified against published data. The results will be deemed unacceptable if the percent error is above 10%. The proposed lifting body glider is solely experimental without any existing flight data or wind tunnel testing. Thus, the initial aerodynamic properties will be calculated at a 30-km altitude, where the flight is expected to begin, with varying angle of attack (α). A glide angle will be determined, and the velocities will be estimated from launch to landing to begin the formation of a flight envelope. At each step in the developmental process, or with any changes to the MATLAB code, the flight envelope from 30 km is recalculated to ensure the vehicle is being developed under the correct flight conditions.

3.1.1 Implementing the Vortex Lattice Method

The velocity field induced by the infinite line vortex can be modeled by integrating finite vortex segments. The velocity of the induced field is dictated by the Biot-Savart Law, as seen in Equation 1.

$$d\vec{q} = \frac{\Gamma}{4\pi}\frac{d\vec{\ell} \times (\vec{r} - \vec{r}')}{|\vec{r} - \vec{r}'|^3} \tag{1}$$

Where $\vec{r}$ is the point where the velocity field is induced, $d\vec{\ell}$ is the vortex vector length, and $\vec{r}'$ is the location of the infinitesimal segment [7]. The velocity is computed by integrating Equation 1. The relationship of the variables is explained in Figure 2.

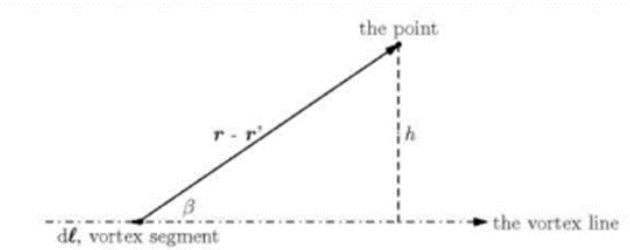


Figure 2. Nomenclature for calculating the induced velocity of a vortex segment [7].

Where the perpendicular distance from the line is $\left|\widehat{\ell \times (\vec{r} - \vec{r}')}\right| = |\vec{r} - \vec{r}'| sin\beta \equiv h$. For a straight line, Equation 1 is integrated to solve for the induced velocity, $\vec{q}$, in Equation 2. Figure 3 shows the corresponding nomenclature for Equation 2.

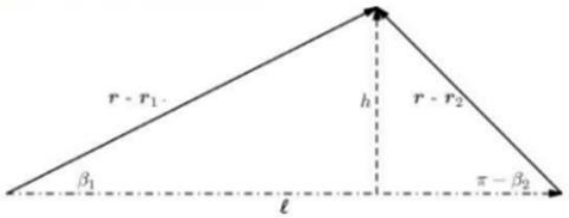


Figure 3. Nomenclature for Equation 2 [7].

$$\vec{q} = \frac{\Gamma}{4\pi} \frac{\hat{\ell} \times (\vec{r} - \vec{r}')}{\left|\hat{\ell} \times (\vec{r} - \vec{r}')\right|^2} (cos\beta_1 - cos\beta_2) \tag{2}$$

Here, the cosines can be expressed in terms of the position vectors such that $(cos\beta_1 - cos\beta_2) = \hat{\ell} \cdot \left(\frac{\vec{r}-\vec{r}_1}{|\vec{r}-\vec{r}_1|} - \frac{\vec{r}-\vec{r}_2}{|\vec{r}-\vec{r}_2|}\right)$.

The VLM sections the mean surface of the wing into an array of panels. Each panel has a horseshoe vortex placed on it with the trailing vortices oriented in the x-direction towards the trailing edge. The method assumes that the far wake vortices are parallel to the oncoming stream because the far vortices have little influence, rather the near vortices are dominant. The bound segment vortex is laid along the quarter-line of each panel. This location is found by linearly interpolating a quarter of the way from the front edge of the panel. The collocation point, the location of the singularity, is similarly found and placed at the three-quarter point by

interpolating two-thirds from the midpoint of the previously calculated bound vortex. The locations of the relevant points are shown in Figure 4.

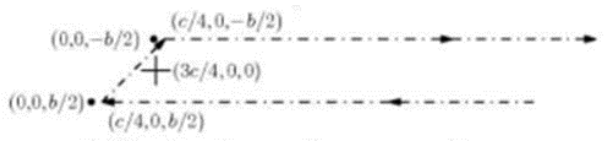


Figure 4. Bound vortex segment and collocation point [7].

Once the locations of the bound vortex segments and collocation points for each panel have been calculated, the boundary condition that the surface is impermeable is applied to calculate the strength, Γ. The boundary condition is enforced by balancing the normal component of the free stream with the normal component of the induced velocity on each panel. Thus, the VLM equation becomes

$$\sum_{j=1}^{n}(\hat{n}_i \cdot \vec{q}_{ij})\Gamma_j = -q_\infty(\hat{n}_i \cdot \vec{\imath}cos\alpha + \hat{n}_i \cdot \vec{\jmath}sin\alpha) \tag{3}$$

where $\hat{n}_i$is the panel normal and α is the angle of attack. The lift is estimated using the Kutta-Joukowsky theorem shown in Equation 4 [7]. Each bound vortex segment of strength Γ_i and span b_i contributes to the total lift force.

$$L = \rho q_\infty \sum_i \Gamma_i b_i \tag{4}$$

The following equations were used to obtain the drag coefficient [4].

$$\alpha_i = \frac{C_L}{\pi AR} \tag{5}$$

$$C_{Di} = \frac{2\alpha_i}{vS}\sum_i \Gamma_i b_i \tag{6}$$

$$C_D = C_{Di} + C_{D0} \tag{7}$$

$$C_{D0} = \sum_{i=1}^{N} \frac{K_i \bar{C}_{f_i} S_{wet_i}}{S_{ref}} \tag{8}$$

Bertin [4] provided an example problem of the VLM which was used for initial verification of the MATLAB code. The wing is flat without taper having an aspect ratio of 5 and a sweep angle of 45°. The span was sectioned into eight panels, four on either side, for a single row of eight horseshoe vortices as shown in Figure 5. Bertin [4] calculates the lift slope to be 3.443, and the developed VLM calculates it to be 3.4442 resulting in an acceptable error of 0.035%.

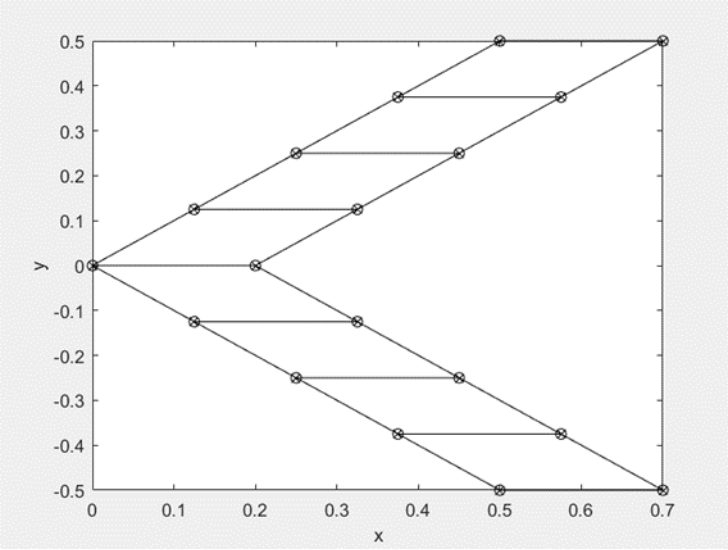


Figure 5. Planform of the Bertin wing

Next, the delta wing from Melin's Tornado program was modeled. The delta wing had no dihedral and a sweep angle of 70°. This test was conducted with a freestream velocity value and served to verify the boundary condition code. The values calculated for the lift, induced drag, lift coefficient, and induced drag coefficient were 339.3491 Newtons, 11.2921 Newtons, 0.1522, and 0.0051, respectively. Compared to Melin's values of 339.1732 Newtons, 11.1225 Newtons, 0.1521, and 0.0050, the percent error was 0.052 %, 1.525 %, 0.045 %, and 1.5164 %, respectively. The resulting vortex strength distribution is shown in Figure 6, where the circles

represent the midpoints of the vortex segments and the aerodynamic forces are the normal vector lines.

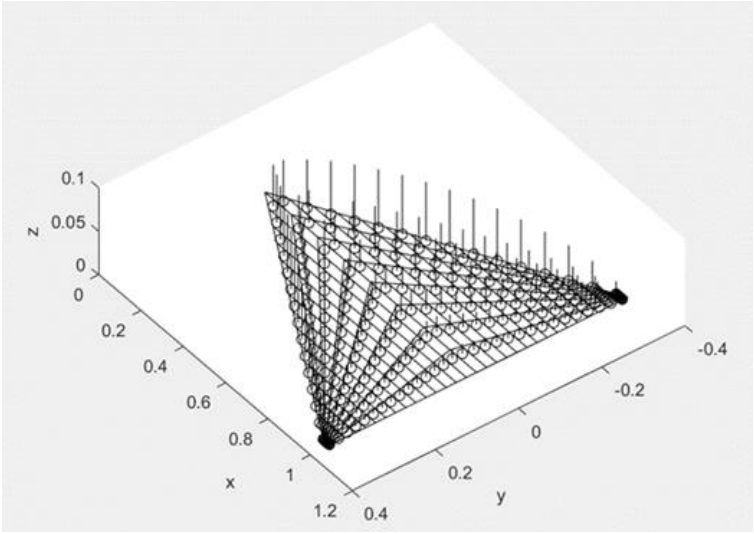


Figure 6. Aerodynamic force vectors acting on the midpoints of the bound vortex segments

Once the fundamentals of the code had been verified, the next step was to attempt more complex geometry. Since the objective of the project is to design a lifting body, the space shuttle was chosen for the initial tests. Boyden and Freeman [8] documented the subsonic characteristics of a 0.0165 scale model of the space shuttle orbiter in a wind tunnel. The scale model had a double-delta planform with 81° sweep on the main fillet and 45° sweep on the main wing. The model had a vertical tail with a rudder which was flared 10° for the basic configuration. The model had a reference area of 0.068 m^2, a length of 0.54076 m, and a span of 0.39259 m. Figures 7 through 9 show the computed values against the documented values for the basic configuration at Mach 0.3. It is evident from the results that the current methodology is sufficient for low angles of attack, between 0 and 10, to effectively estimate the lift coefficient.

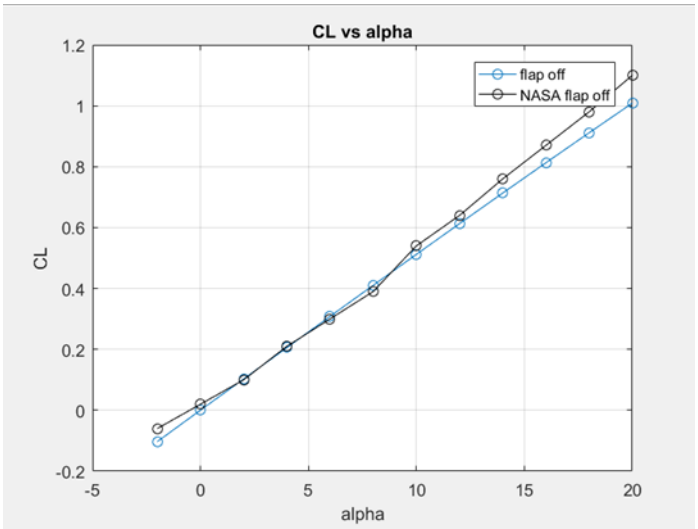


Figure 7. Space Shuttle CL vs α

It is evident from the results that the current methodology is sufficient for low angles of α, between 0° and 10°, to effectively estimate the lift coefficient. However, as α becomes negative or is greater than 10°, deviation from the experimental data occurs. The largest error is at α equal to -2° and 0° and is greater than 10%. The rest of the data are within the allowable 10% error.

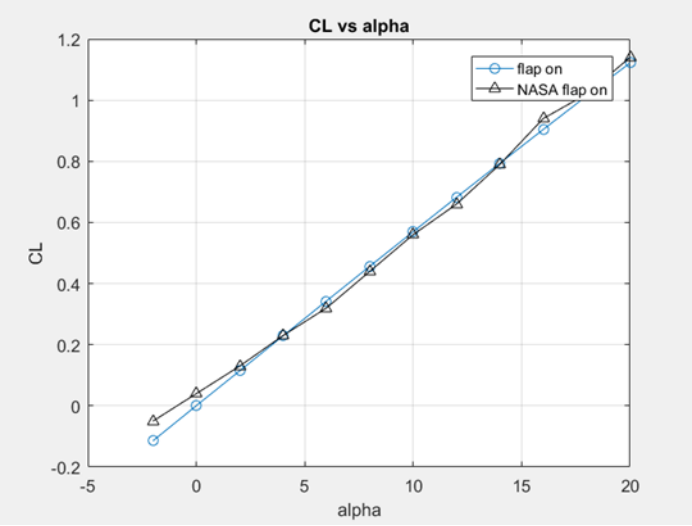


Figure 8. Space Shuttle CL vs α including the body flap contribution

The results with the body flap added to the model, Figure 8, exhibit similar behavior. As α approaches 0° and negative values, the data begin to deviate. As with the previous results, the

error for α at the values of -2° to 2° exceeds the maximum allowable error while the values of α above 2° fall within the acceptable 10 % error.

The drag coefficient data, plotted in Figure 9, show that the data significantly deviate as α increases. After α = 8°, the slope of the curve is drastically different. This is likely due to an incorrect estimation of the parasite drag. It is also probable that the drag due to lift from the rudder contribution is incorrectly calculated as the α increases. More investigation was needed to determine how to properly model deflected control surfaces as the angle of attack for the body changes as well as the parasite drag value. It is unlikely that the fundamental VLM code is incorrect, because it works very well for simple wings, rather that the geometric modeling of the vehicles is causing the deviations.

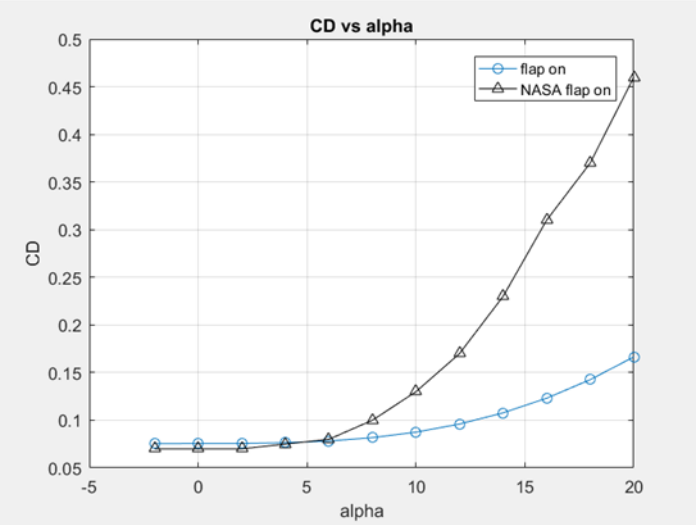


Figure 9. Space Shuttle CD vs α

3.1.2 Adjusting for Two-dimensional Effects

Wing sections can also be modeled using panel methods that distribute the sources and vortices around the profile rather than along the mean line. The profile is broken up into a chain of segments, each of which has a uniform vortex strength distribution [7]. As with the 3D method, the unknown vortex strengths are calculated by the boundary conditions of each panel

applied at their collocation points. The Kutta-Joukowsky theorem is again used to determine the vortex strengths of the panels.

A vortex is a source with an imaginary strength, thus the 2D VLM treats the source and vortices together. A complex source of strength $Q + i\Gamma$ induces a velocity $w(z) = \frac{Q+i\Gamma}{2\pi(z-z')}$. Integrating this along the segment results in $w(z) = \frac{\ln z - \ln(z-1)}{2\pi}$. Setting $\zeta = \frac{z-z_0}{z_1-z_0}$, the corresponding velocity in the z-plane is shown in Equation 9.

$$w(\zeta) = \frac{\ln\frac{z-z_0}{z_1-z_0} - \ln\frac{z-z_1}{z_1-z_0}}{2\pi(z_1-z_0)} \tag{9}$$

The velocity induced at the collocation point becomes $\omega_{ij} \equiv \frac{\ln\frac{z_i^{(c)}-z_j}{d_j} - \ln\frac{z_i^{(c)}-z_j-d_j}{d_j}}{2\pi d_j}$ where $z_i^{(c)}$ is the collocation point and d_j is the complex length of the panel. The induced velocity is subsequently corrected so it is oriented in the proper direction, as seen in Equation 10, where λ_i is the slope of the panel.

$$\omega_{ii} = |\omega_{ii}|e^{-i\left(\lambda_i - \frac{\pi}{2}\right)} \tag{10}$$

The 2D VLM equation becomes Equation 11, where $\Re$ represents the corrected tangential component since the panels are traced counterclockwise from the trailing edge. The lift coefficient is subsequently calculated with Equation 12. Equations 13 and 14 show calculations for C_p and C_d, respectively.

$$\Re\sum_{i=1}^{n} e^{i\lambda_i}\left(\sum_{j=1}^{n}\omega_{ij}Q_j + \left[\sum_{j=1}^{n}\omega_{ij}i\right]\Gamma\right) = -\Re q_\infty\left\{e^{i(\lambda_1-\alpha)} + e^{i(\lambda_n-\alpha)}\right\} \tag{11}$$

$$Cl = \frac{2\Gamma}{q_\infty c} \tag{12}$$

$$C_p = 1 - \left(\frac{v}{U}\right)^2 \tag{13}$$

$$C_d = \sum -C_{p,lower} sin\lambda d_i + \sum C_{p,upper} sin\lambda d_i \tag{14}$$

The next configuration tested was the HL-20 lifting body. When the traditional VLM code was implemented for this geometry the lift coefficient slope was much too high. It was believed that the use of the mean line to calculate the aerodynamic forces is insufficient to calculate the forces for a lifting body fuselage as well as the blunt trailing edge of the lifting body. A 2D VLM was developed to account for the upper and lower surfaces of the lifting body rather than using the mean surface. The method was developed from McBain [7] and verified using examples from the book. Figure 10 is an image of the velocity field around the NACA 2412 airfoil at an 8° angle of attack. The calculated lift coefficient was just slightly higher than the reported coefficient from the lifting line theory with an error of 4.35 %.

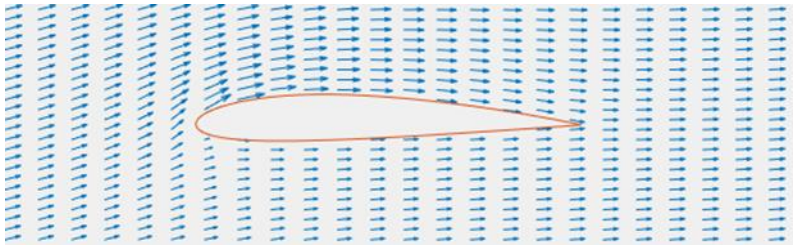


Figure 10. Velocity field around the NACA 2412 airfoil computed in MATLAB.

Huffman et. al. [9] conducted subsonic wind tunnel tests on the HL20 vehicle and derivative models which had modified bodies. The report documented the body alone configurations, i.e., the lifting body without any wings attached, with detailed model dimensions at a Mach number of 0.3 and Reynolds number per foot of roughly 1.8 x 10^6. The HL-20A-1 model was used to develop the following VLM which is to be used on the blunt-ended lifting body fuselage. Due to the blunt trailing edge of the body, it was unclear whether the 2D method

would even be suitable for estimating the aerodynamic forces. The 2D method was implemented at three cross sections along the span, and then mirrored in the x-plane for a total of 6 cross sections. The subsequently calculated force values were summed over the span. The coordinates were taken from Solidworks sketches that were created at specific locations along the span. Figure 11 shows a sketch of the innermost cross section and the resulting C_p distribution at α = 0°. The result was compared to XFOIL data calculated for the same set of coordinates, shown in Figure 12. The calculated lift coefficient using the VLM was 0.0431, which when compared to the XFOIL value of 0. 0408 resulted in an error of 5.66 %. It is likely that this error is due to the spacing of the airfoil coordinates in MATLAB and can probably be remedied.

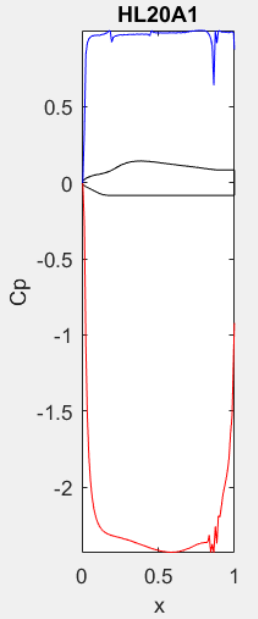


Figure 11. HL-20-A1 C_p.

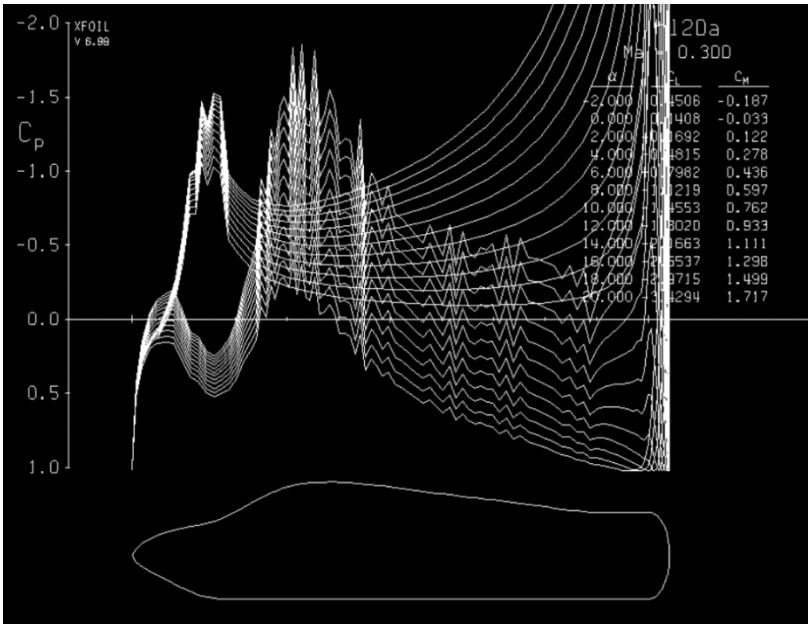


Figure 12. HL-20-A1 XFOIL results at M=0.3, Re = 1.8 x 10^6.

The aerodynamic forces were then calculated for an α range of -2° to 20°. Figure 13 shows the variation of the lift coefficient for the different methods against the experimental data. The 2D method accurately estimated the value at α = 0°, but the lift-curve slope was too low. Conversely, the 3D method failed to accurately estimate the value at α = 0° and the lift slope was

too high. Thus, a weighted average was used to average the two methods, which is shown as the VLM line on the plot. The weighted average was the 3D coefficient added to twice the 2D coefficient, summed and divided by two. The result is a closely correlating line with the maximum error at α = -2°.

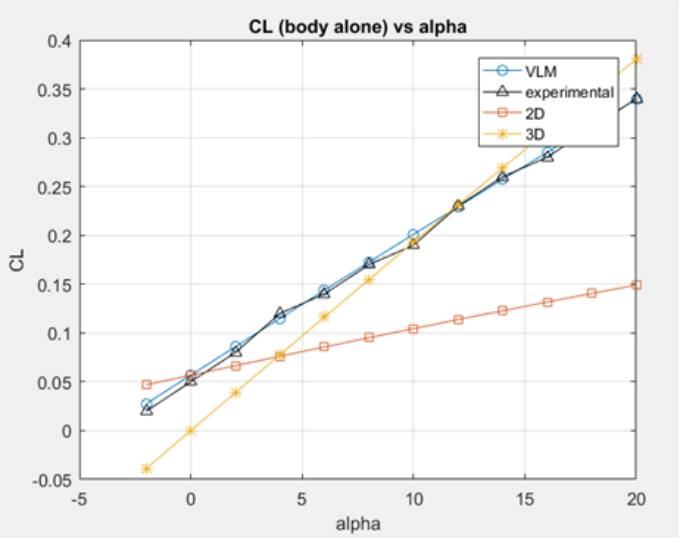


Figure 13. HL-20-A1 variation of C_L with α

Figure 14 shows the HL-20-A1 body alone and full configuration, where the updated VLM data include both the two and three-dimensional contributions and the experimental data. It was determined that including the two-dimensional characteristics for both the body and wings was most accurate.

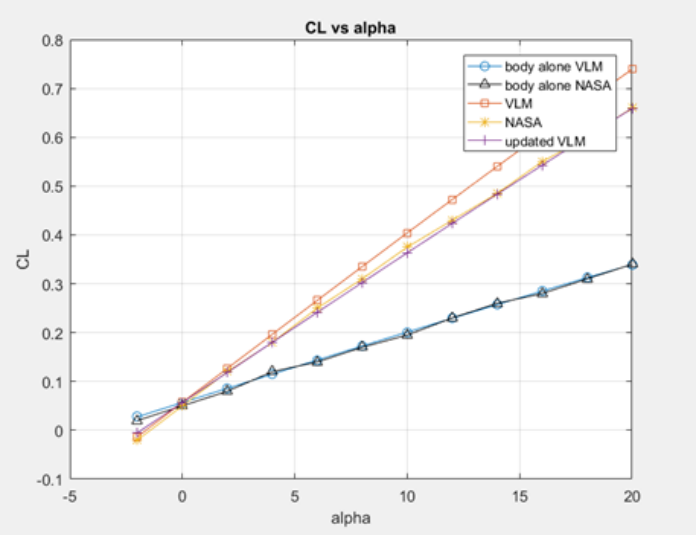


Figure 14. HL-20-A1 body alone and with wings.

3.2 Modeling the Lifting Body Geometry in MATLAB

3.2.1 Initial Conditions

After preliminary verification of the computational model was completed, the proposed lifting body glider was modeled. In the same fashion as the HL-20-A1, measurements were taken by hand and imported into Solidworks to sketch the cross-sectional geometry of the vehicle. The coordinates were again scaled and sectioned to 100 points along the x-axis with corresponding upper and lower values. The glider has a length of 1 m with a span of 0.8334 meters with a wing dihedral of 17.67°. The glider reference values are described in Table 1 and are shown in Figure 15.

Reference area, S_{ref} ..... 0.2802 m^2
Reference span, b_{ref} ..... 0.4718 m
Reference length, d_{ref} ..... 1.0367 m

Table 1. Reference values for lifting body glider

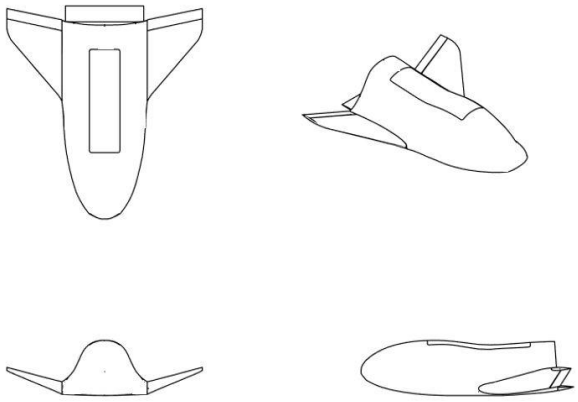


Figure 15. Lifting body glider.

The glider was initially tested at an altitude of 30 km, where the balloon was expected to pop, with a speed of 0.01 m/s to get an estimate of what the velocity might be at sea level. The geometry used in the preliminary VLM was simplified to omit camber, under the assumption the methodology could sufficiently estimate the forces without including camber in the three-dimensional method. Figure 16 is a plot showing the midpoints of the bound vortex segments and vortex strengths gamma, Γ, for this configuration.

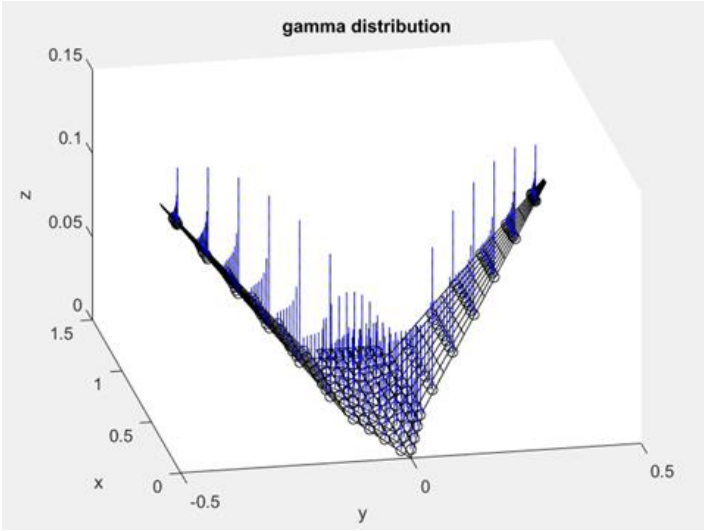


Figure 16. Aerodynamic force vectors acting on the midpoints of the bound vortex segments.

The glide velocity was determined using the maximum lift over drag (L/D) value and the calculated glide angle (θ_{min}) using equations 14 and 15. Equation 16 was used to calculate the velocity as the glider descends from the drop height. Figure 17 shows the relationship between velocity and altitude as the glider descends. At sea level, the initial velocity at standard temperature and pressure (STP) was estimated to be 12-15 m/s for θ_{min} of about -1.5°.

$$tan\theta_{min} = \frac{1}{\left(\frac{L}{D}\right)_{max}} \tag{14}$$

$$v_{\left(\frac{L}{D}\right)_{max}} = \left(\frac{2}{\rho_\infty}\sqrt{\frac{K}{C_{D,0}}}\frac{W}{S}\right)^{\frac{1}{2}} \tag{15}$$

$$\left[v_{\left(\frac{L}{D}\right)_{max}}\right]_Z = \left[\frac{(\rho_\infty)_{30km}}{(\rho_\infty)_Z}\right]^{\frac{1}{2}}\left[v_{\left(\frac{L}{D}\right)_{max}}\right]_{30km} \tag{16}$$

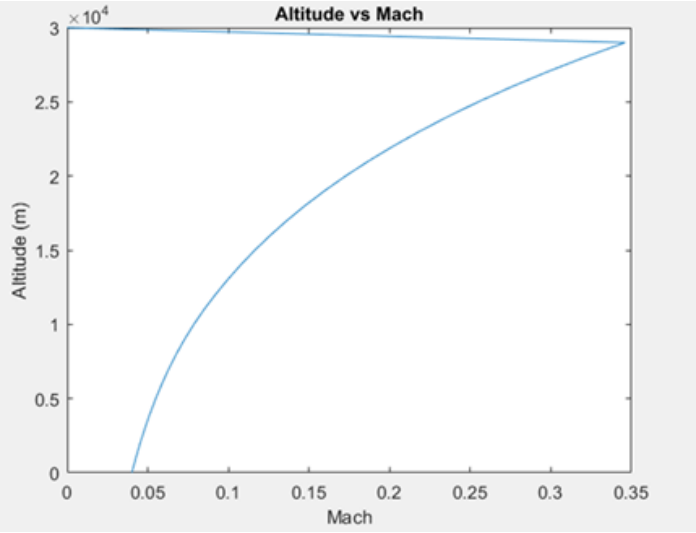


Figure 17. Mach as a function of altitude during descent.

3.2.2 Solidworks Flow Simulation

Solidworks has the capability to perform flow simulations of an object within a 3D volume and was used as a verification tool for the aerodynamic MATLAB code. The solver can

be used to obtain lift, drag, and side force values, among many other relevant features such as pressure and velocity distributions. The margin of error is unknown and therefore the data are not considered to be an absolutely accurate descriptor of the glider aerodynamics. However, it can be used to compare results from different methodologies, i.e., if both MATLAB and Solidworks calculate the same result, it is likely to be relatively accurate. Additionally, the reference coordinate system in Solidworks Flow Simulation is different from that of the body coordinate frame for an aircraft. This caused some error within the calculations that was intuitively adjusted to reflect the differences between what was calculated in Solidworks and what is typical for an aircraft.

The flow simulations were conducted on the 3D model of the glider developed in Solidworks under STP at 15 m/s. The glider was rotated about the center of mass through an angle of attack and sideslip range using the pattern features in Solidworks. Each simulation used an initial mesh density of 6, which has 74 cells in the x-direction, 64 cells in the y-direction, and 121 cells in the z-direction. The initial mesh is shown in Figure 18.

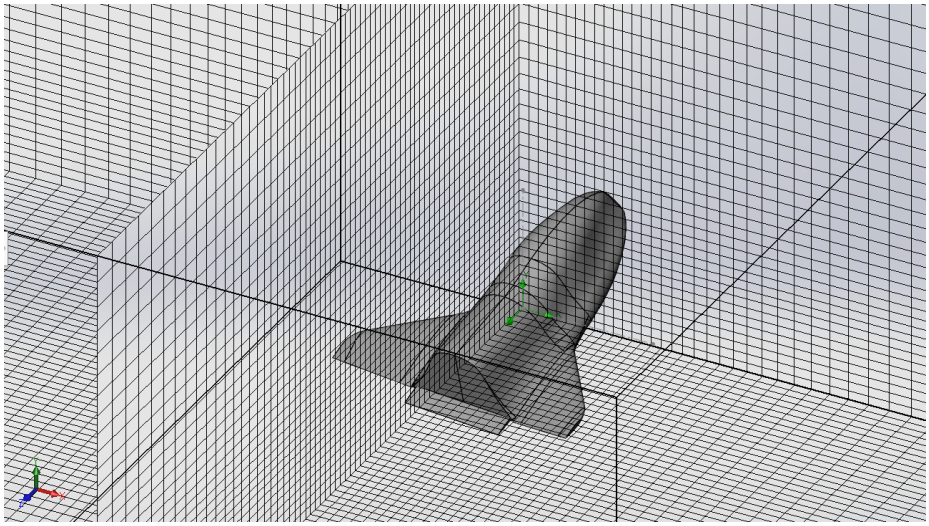


Figure 18. Initial glider mesh in Solidworks.

In addition to the high-density mesh control, the advanced channel refinement setting was initialized so that Solidworks can increase the number of iterations and refine the mesh density where needed. This setting improves computation times, rather than having a very dense initial setting throughout the computational volume. It is important to note that changing the initial mesh settings to higher or lower densities will change the results. However, because there were no flight data when the Solidworks Flow Simulations were being conducted, it was unknown which mesh settings most accurately modelled the glider aerodynamics. Thus, the decision was made due to computation times. With the current mesh settings, the force calculations in three directions took 40 minutes to an hour to converge. Adding in the moment calculations took even longer to converge. Additionally, halfway through this research the Solidworks license was updated and the same settings in 2017 did not yield the same results in 2018. The 2018 results were slightly higher than 2017, so when the transition to the new license was made the mesh density was reduced so that the data would remain relatively consistent. An image of the glider with the calculated streamlines using the flow simulation is shown in Figure 19.

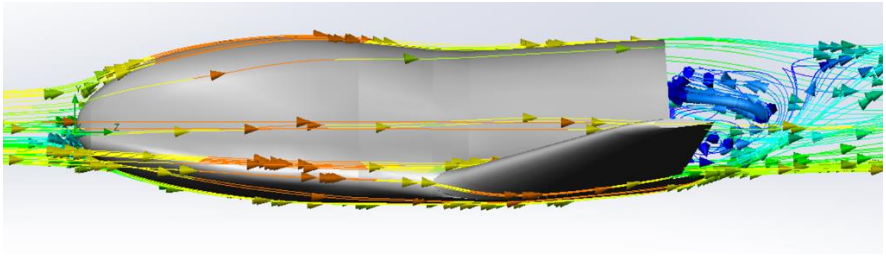


Figure 19. Plot of streamlines calculated in Solidworks Flow Simulation at v = 15 m/s, $\alpha = 0°$.

The primary issue with using Solidworks as a verification method is that Solidworks calculates the side force opposite to the typical sign convention causing the side force results to be opposite from what they should be. While this is not a particularly serious problem in calculating the forces, the difference in reference coordinate systems may lead to large

inaccuracies when calculating the moments. The pitching and yaw moments calculated as was expected, but the rolling moment flipped slope as α increased. Further investigation into the moment calculations revealed inconsistencies in the trends with respect to changing α and β. With increasing α, the pitching moment should decrease, not increase as it was in Solidworks. The yaw and rolling moment behaviour is better described with changing β but each had significant changes with α, which is not to be expected. A center fin was added to the model to determine whether it would improve the lateral moments, but instead of stabilizing the rolling moment it destabilized it even further, which is completely contrary to aerodynamic design standards. At this point it was determined that while Solidworks could effectively estimate the forces, in the lift and drag directions at least, that it would not be used for moment verification.

3.2.3 Developing the Computational Model

The first attempt at verifying the MATLAB results to the Solidworks Flow Simulation results was unsuccessful because the % error between the sets of data was over 10 %. Upon further investigation, it was determined that XFoil better predicted the two-dimensional aerodynamic characteristics for the body. It is likely that XFoil is the better prediction method because the program required a larger number of coordinates distributed around the profile and is therefore more accurate for the glider-specific geometry. A better methodology for the traditional VLM was identified as well. It was found by trial and error that including the mean camber line of the fuselage improved the accuracy of the force estimation. The best results occurred when averaging the values calculated for the fuselage with and without camber for the 3D contribution. For this new method, a weighted average was unnecessary, simply taking the average of the 2D and 3D contributions proved to be sufficiently accurate. While there is still some uncertainty with computational modeling without flight data to act as a baseline, i.e., there are many ways to

combine the data to obtain the desired results, it is much more realistic to average the 2D and 3D data than to take a weighted average; thus, the updated code was accepted to be more accurate. After the computational model was updated to use XFoil and include camber, the % error between MATLAB and Solidworks fell below 10%. At this point, an assessment of $(L/D)_{max}$ and static longitudinal stability was performed. The initial configuration had an acceptable $(L/D)_{max}$ but the pitching moment (C_m) was bordering on becoming unstable, so a −2° incidence angle was introduced to the wings. This caused the positive pitching moment to improve, while only slightly sacrificing lift. The glide angle and velocity results were recalculated to reflect the changes in the code, resulting in a velocity at STP of 15-18 m/s. Figures 20 and 21 show the changes to the MATLAB VLM code compared to the Solidworks Flow Simulation data for C_L and C_D, respectively. Figure 22 shows the longitudinal characteristics after the code update. Additionally, the stall limit was calculated for the anticipated flight path and is shown in Figure 23.

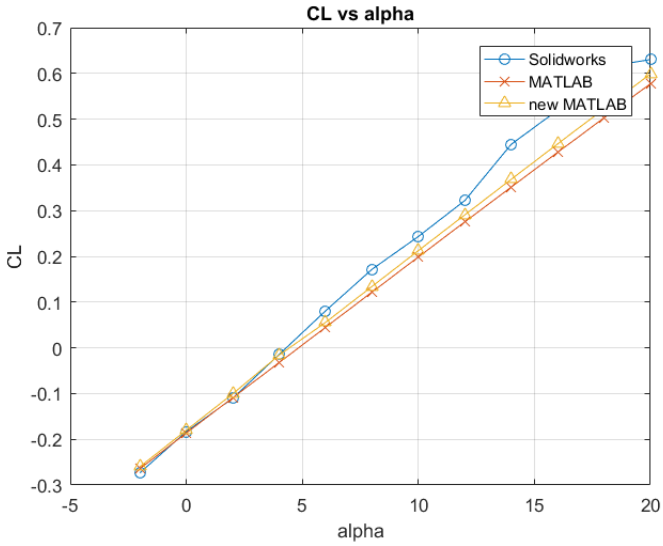


Figure 20. C_L changes with code update to include XFOIL and 3D camber.

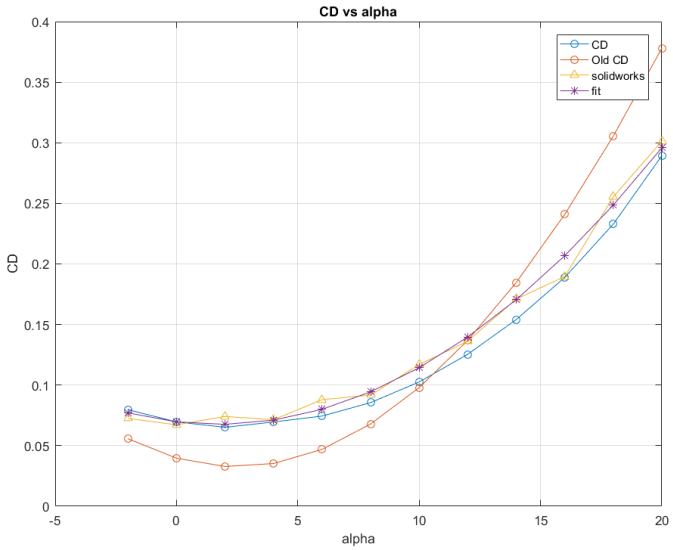


Figure 21. C_D changes with code update to include XFOIL and 3D camber.

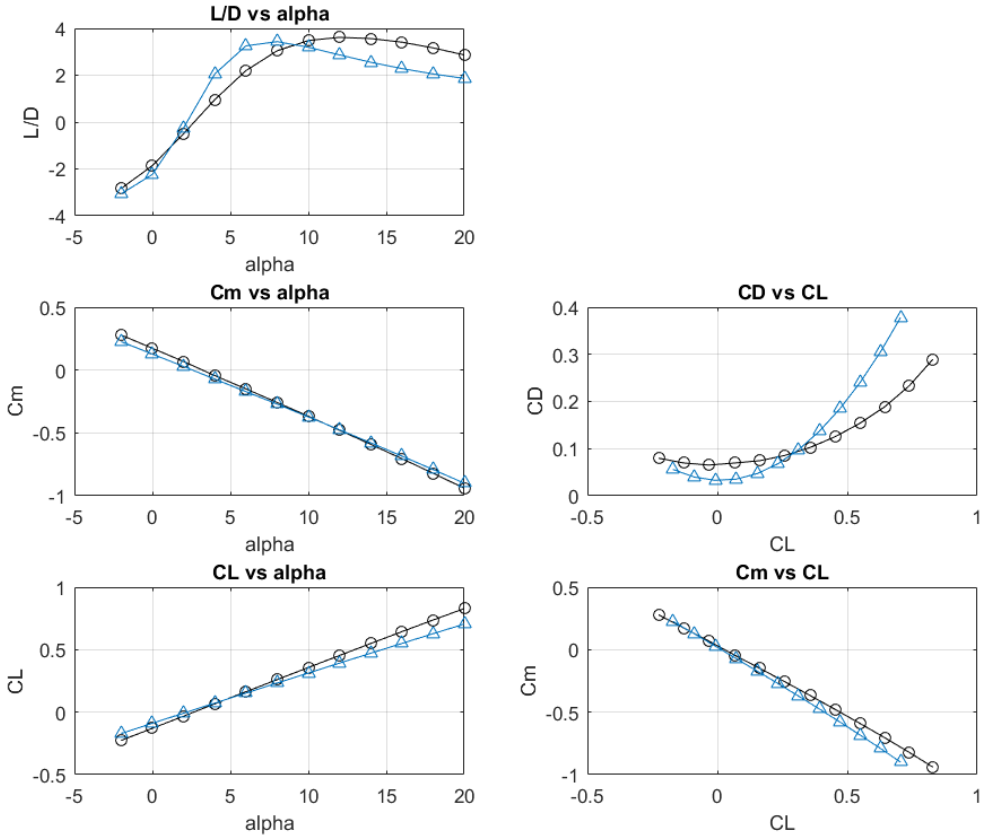


Figure 22. Changes in longitudinal aerodynamic characteristics between the preliminary (black) and updated (blue) VLM code

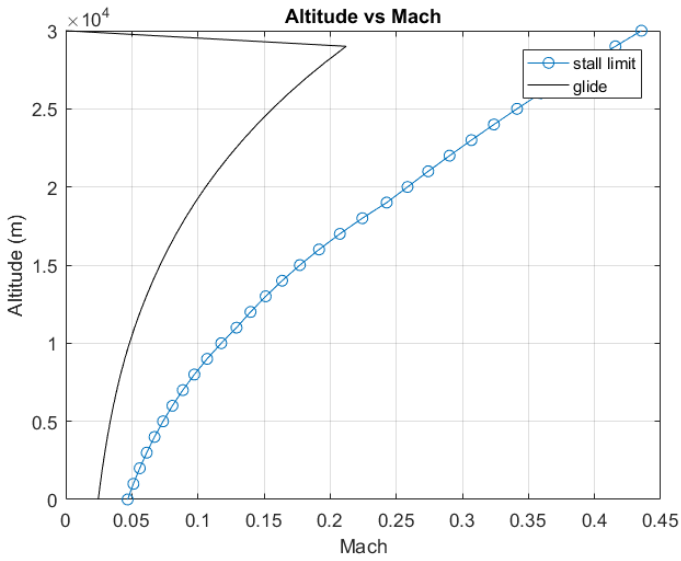
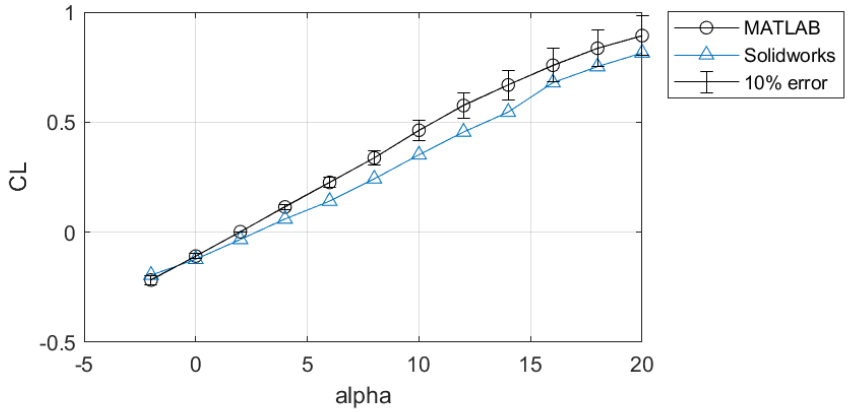


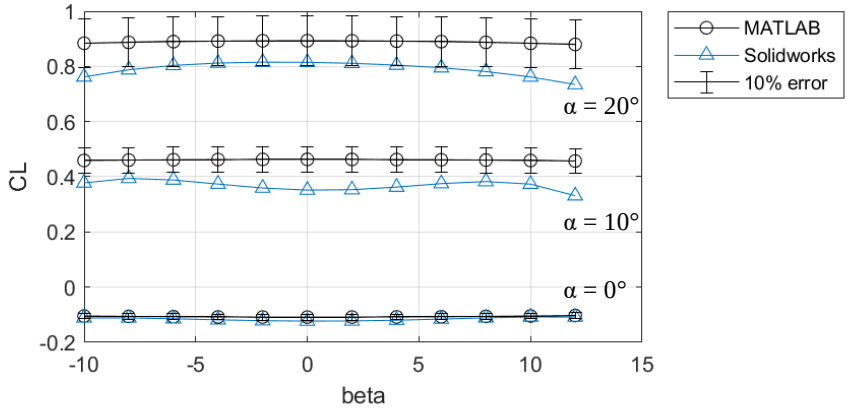
Figure 23. Glide velocity and stall limit.

The basic configuration was tested under an angle of attack range from −2° to 20° and an angle of sideslip range from −10° to 12° at a velocity of 15 m/s. The computational method was subsequently updated to include force measurement in all three axes, using the traditional method outlined in Melin [5]. Using this method, the vortex strengths are multiplied by their respective panel unit vectors in the x, y, and z array indices and incorporated into the boundary condition function. The downwash is separated into its x, y, and z components and multiplied by gamma to produce the inwash which is subtracted from the freestream velocity used to calculate the boundary condition. It is of note that Melin's method of incorporating the inwash into the force calculations resulted in somewhat more accurate results. The cross product of the gamma vectors and oncoming velocity is calculated and multiplied by the density to obtain the force per panel. The result is then multiplied by the respective panel normal to determine the force per panel in the x, y, and z directions and subsequently summed to obtain the total forces. The three-dimensional VLM method still did not accurately predict the aerodynamic forces, thus averaging between the two and three-dimensional methods was still necessary to bring the force values within the allowable error of 10%. XFoil does not predict lateral data, therefore Melin's method

was used to calculate the lateral characteristics of the glider under the assumption it may be more accurate, although the 2D contribution is lacking. The computational model fundamental code can be found in Appendix A. Figures 24 – 29 plot the Solidworks data against the MATLAB data. The moment calculations in Solidworks are assumed to be incorrect, as previously mentioned, but are shown in the figures for reference.

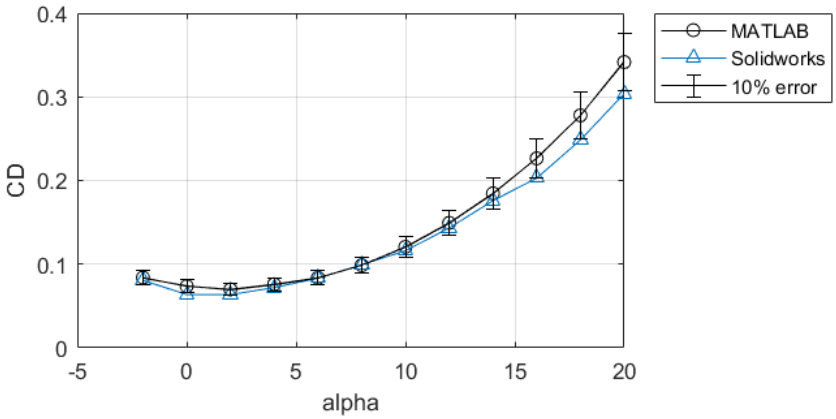


(a) C_L vs. α at $\beta = 0°$

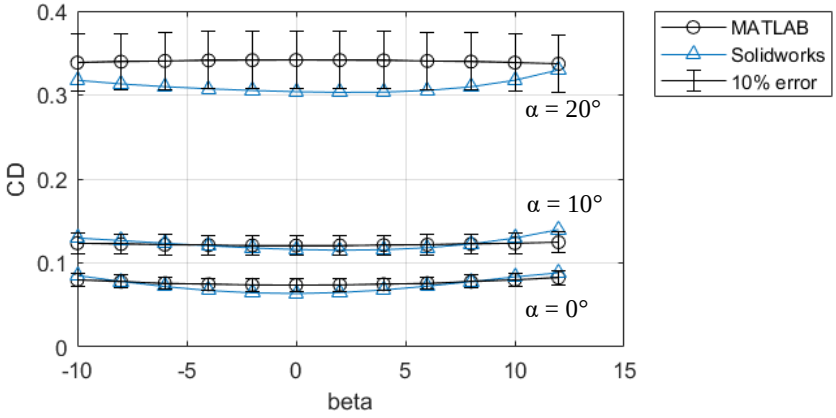


(b) C_L vs. β at $\alpha = 0°$

Figure 24. Lift coefficient C_L for basic configuration

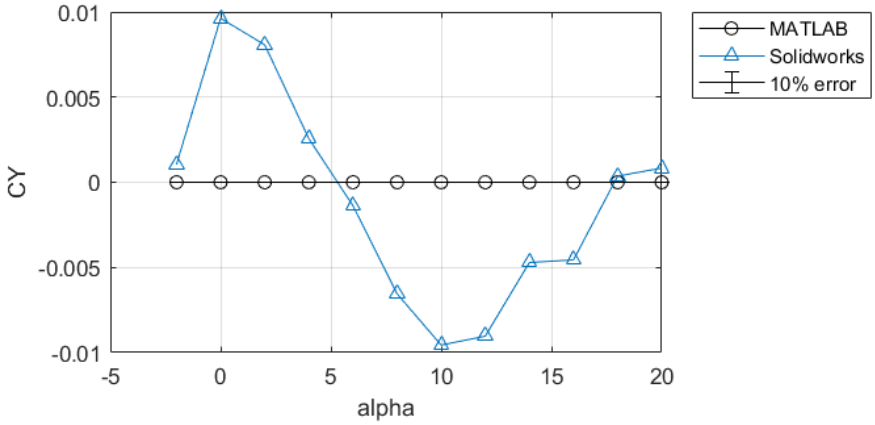


(a) C_D vs. α at β = 0°

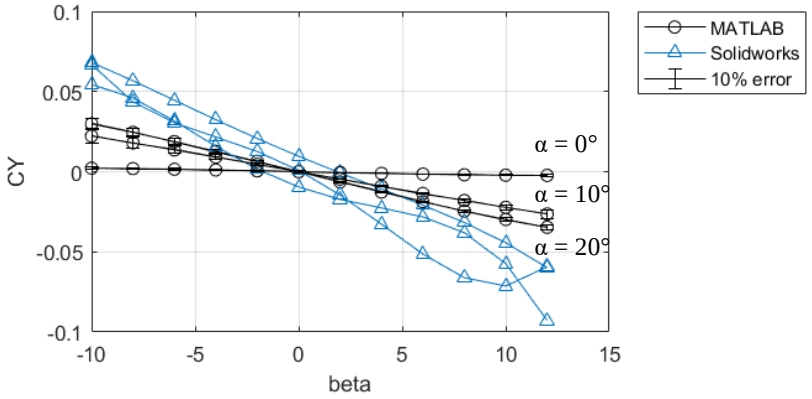


(b) C_L vs. β at α = 0°

Figure 25. Drag coefficient C_D for basic configuration

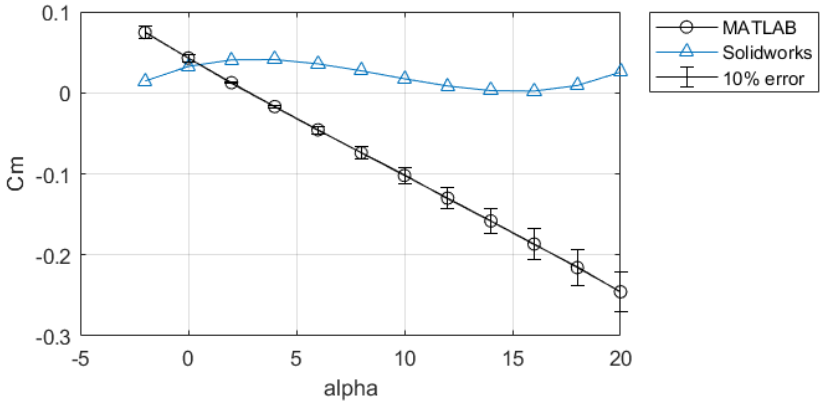


(a) C_Y vs. α at β = 0°

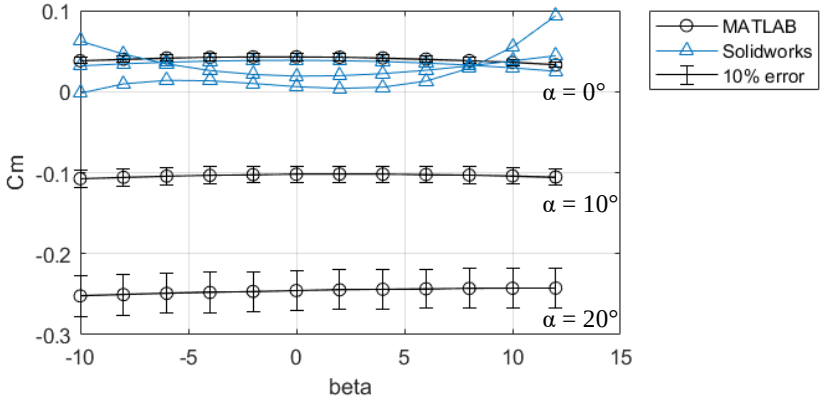


(b) C_Y vs. β at α = 0°

Figure 26. Side force coefficient C_Y for basic configuration

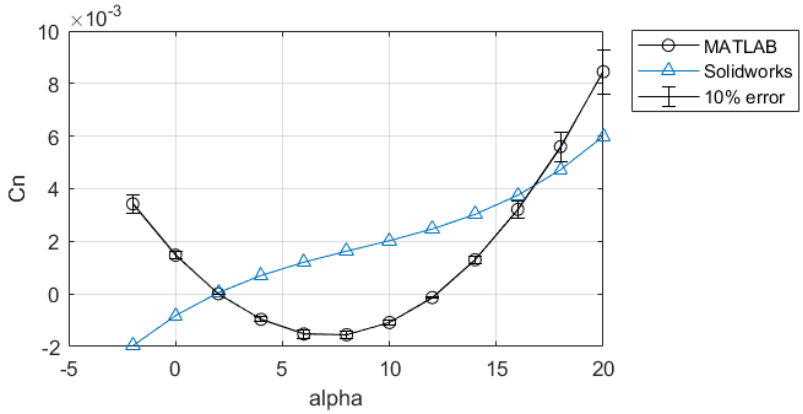


(a) C_m vs. α at β = 0°

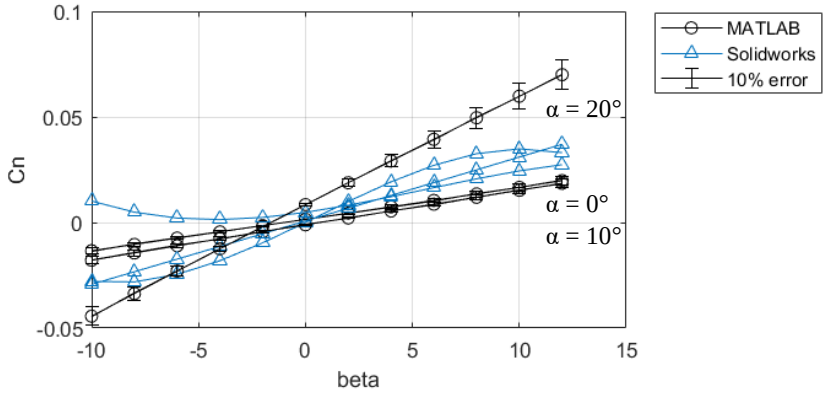


(b) C_m vs. β at α = 0°

Figure 27. Pitching moment coefficient C_m for basic configuration

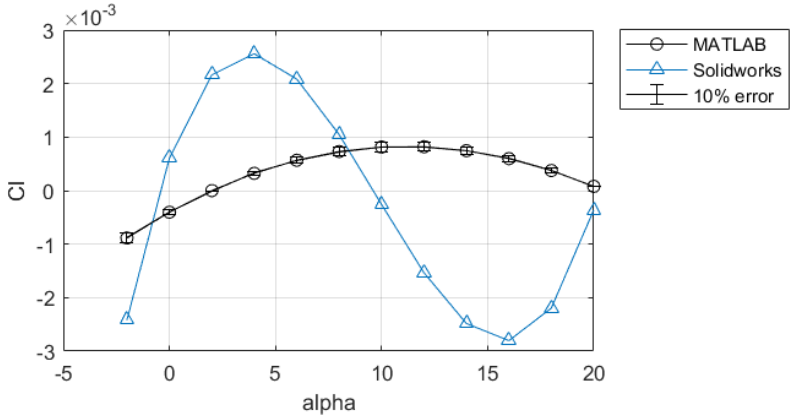


(a) C_n vs. α at β = 0°

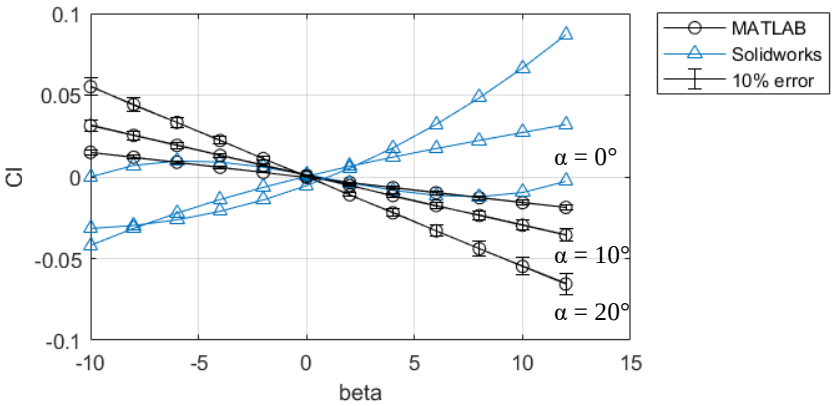


(b) C_n vs. β at α = 0°

Figure 28. Yawing moment C_n for basic configuration



(a) C_l vs. α at β = 0°



(b) C_l vs. β at α = 0°

Figure 29. Rolling moment C_l for basic configuration

It is likely that flight testing may reveal values different than what was predicted, but the MATLAB and Solidworks results show reasonable agreement so they were accepted. While the force values are within 10% error of the Solidworks data, it is possible the aerodynamic moment results will have a higher error due to the unknown 2D contributions for the side force, roll and yaw moments. However, the aerodynamic moment coefficients are expected to accurately indicate the stability of the glider. The side force and rolling moment coefficients are not expected exhibit significant changes with angle of attack and were modelled using a linearly fitted line to simplify the flight simulation. The linear fits are plotted in Figures 30 and 31.

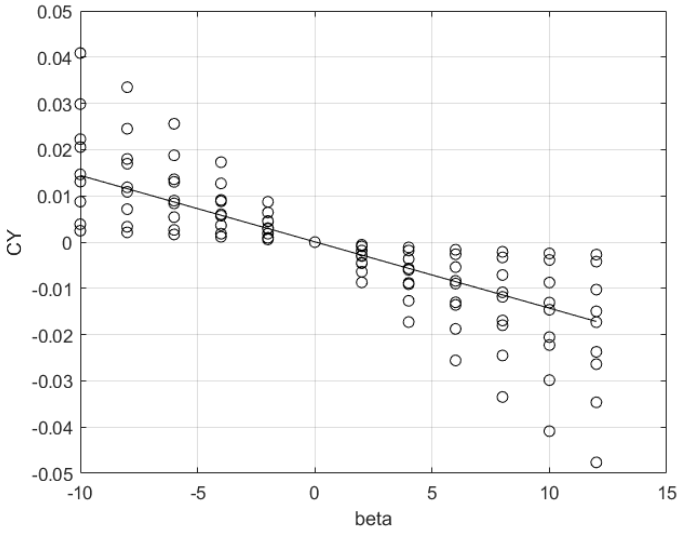


Figure 30. C_Y vs. β for α = -2° to 20°

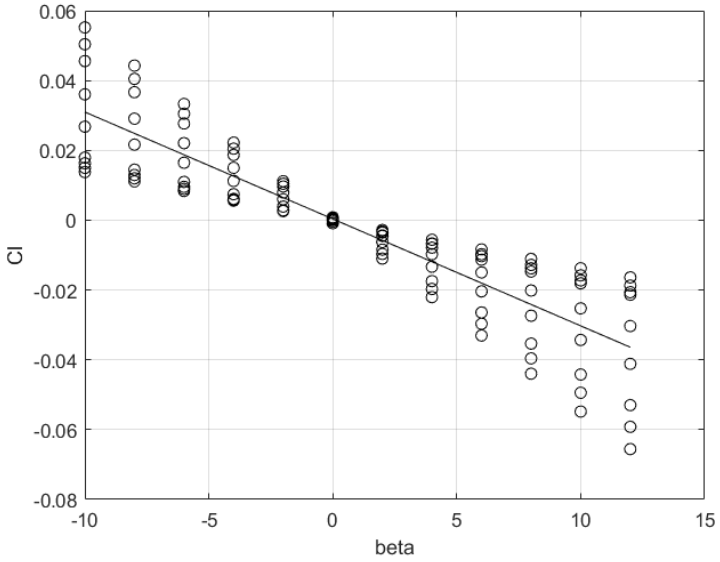


Figure 31. C_l vs. β for α = -2° to 20°

The forces used for the simulation were the body forces, F_X, F_Y, and F_Z, which were found using the wind-to-body transformation matrix. Their calculated values can be found in Tables B1 – B4 in Appendix B, as well as the moments. The moment reference point was located at 57% of the body length along the x-axis. The aerodynamic forces and moments were converted to the nondimensional coefficients using equations 17 – 22.

$$C_X = \frac{F_X}{qS_{ref}} \tag{17}$$

$$C_Y = \frac{F_Y}{qS_{ref}} \tag{18}$$

$$C_Z = \frac{F_Z}{qS_{ref}} \tag{19}$$

$$C_l = \frac{M_X}{qS_{ref}b_{ref}} \tag{20}$$

$$C_m = \frac{M_Y}{qS_{ref}\bar{c}} \tag{21}$$

$$C_n = \frac{M_Z}{qS_{ref}b_{ref}} \tag{22}$$

The control surface contributions were calculated by deflecting the appropriate panels in the MATLAB computational model. Deflections for symmetric ailerons, differential ailerons, and the body flap were computed. The effects of body flap deflection, δ_{bf}, symmetric wing flap deflection, δ_e, and differential wing flap deflection, δ_a, are shown in Figures 32 – 34, respectively. The corresponding values are listed in Tables B5 – B7 in Appendix B, where the values are divided by the degree of deflection for incorporation into the Simulink model.

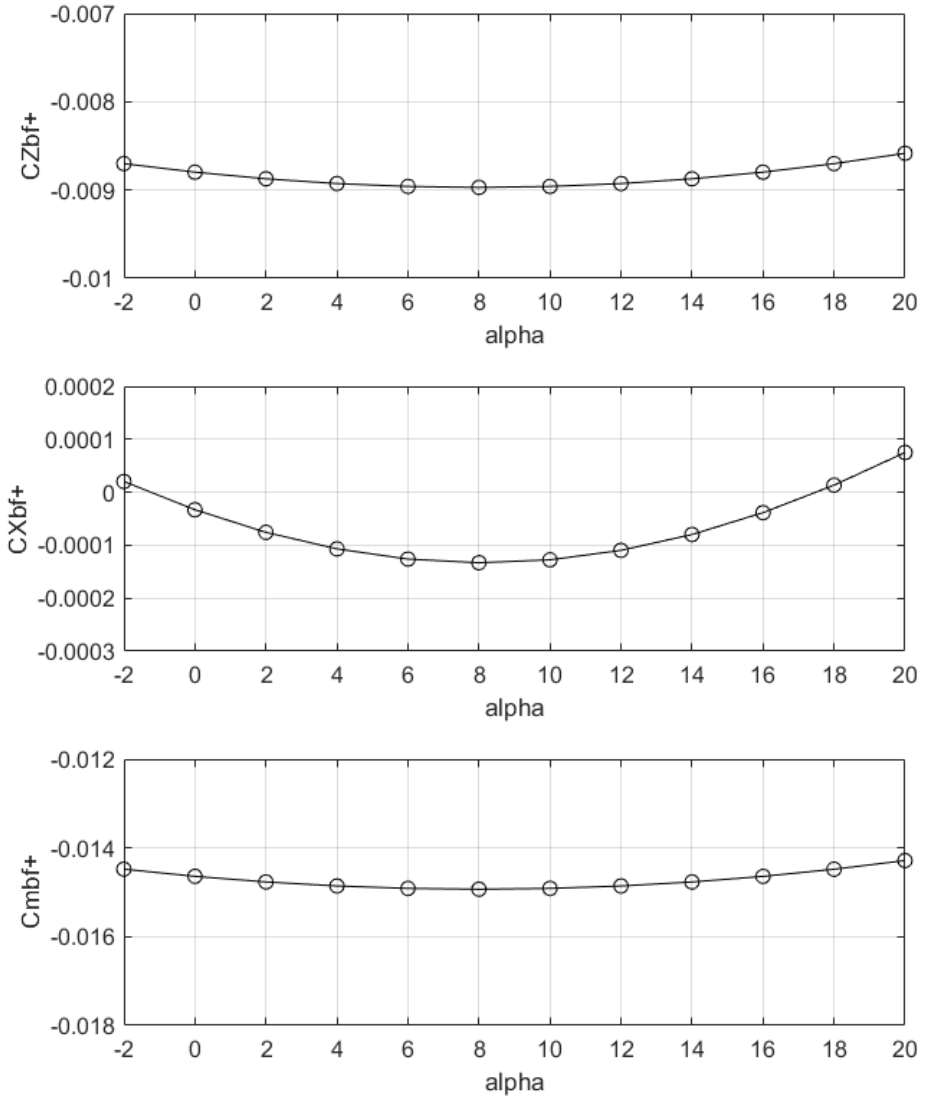


Figure 32. Effect of positive body flap on longitudinal aerodynamic characteristics.

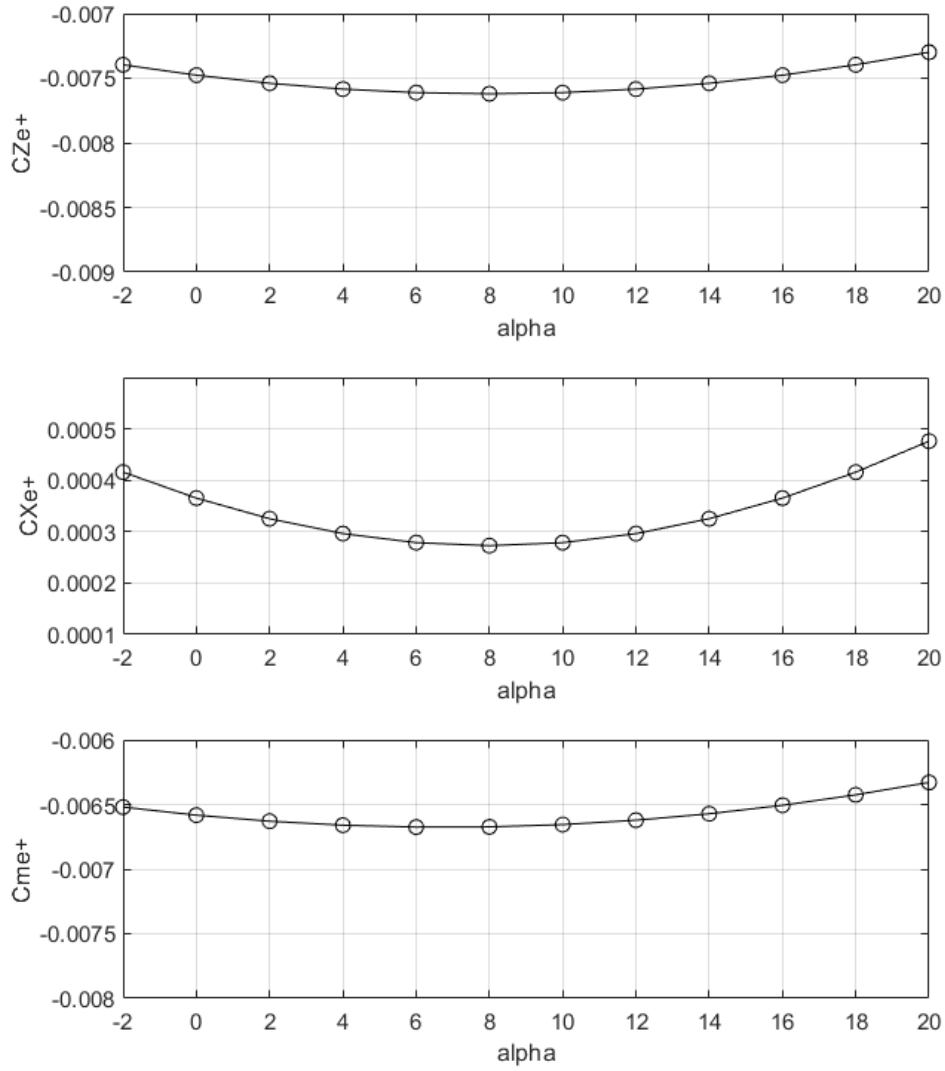
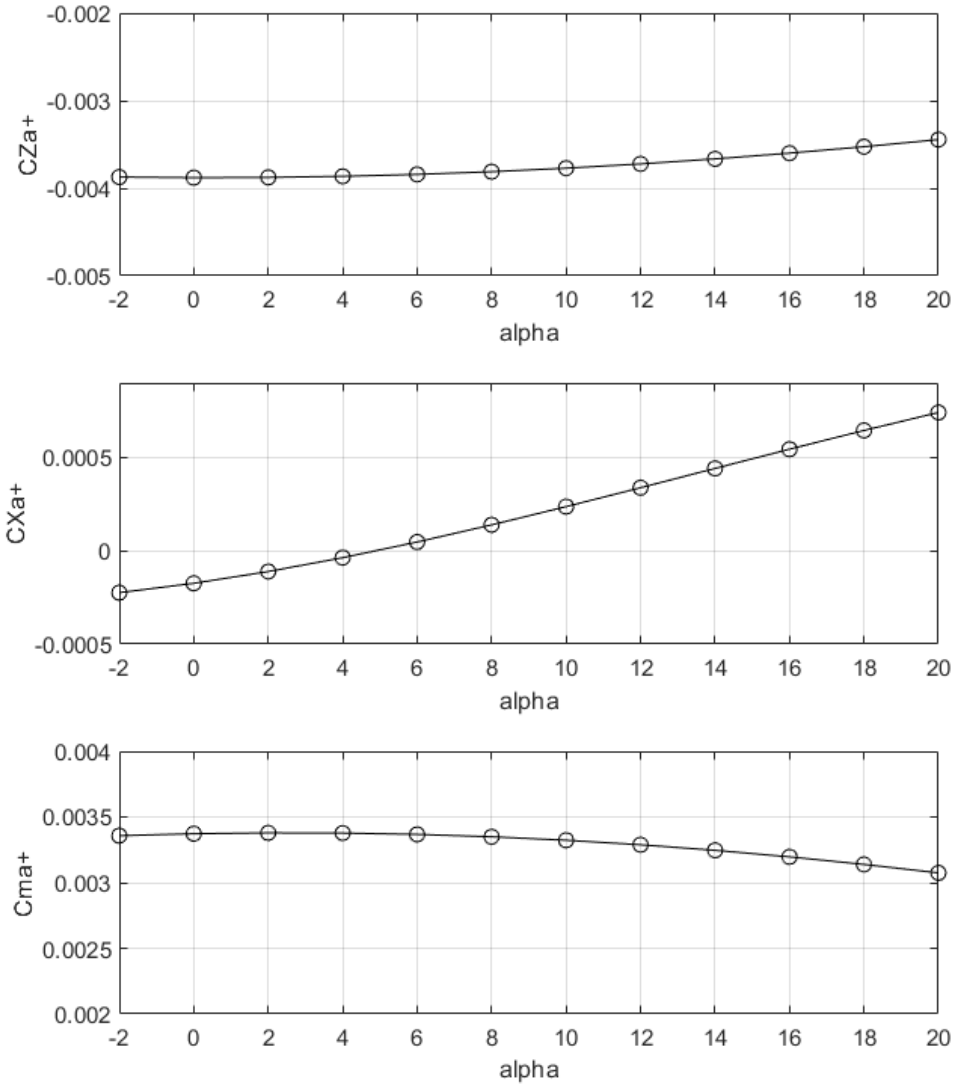
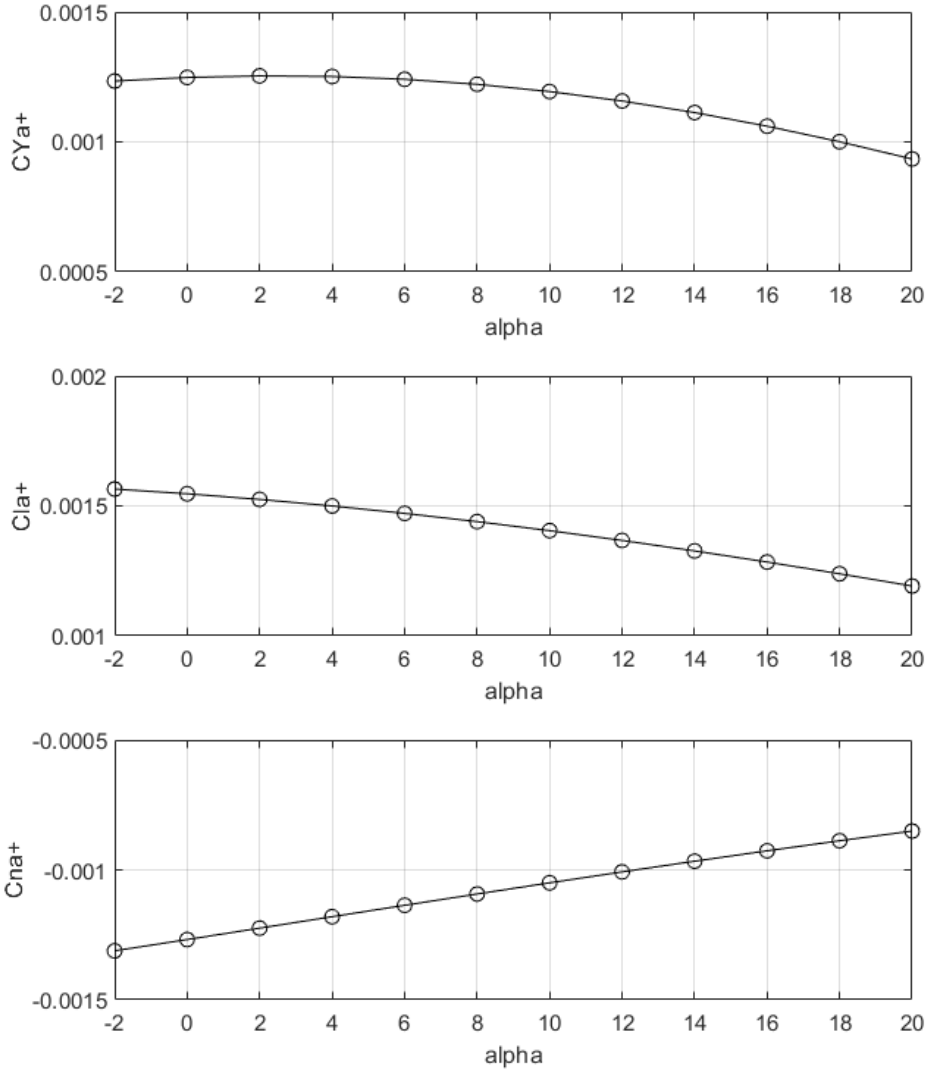


Figure 33. Effect of symmetric wing flaps on longitudinal aerodynamic characteristics.



(a) Longitudinal

Figure 34. Effect of differential wing flaps on aerodynamic characteristics.



(b) Lateral

Figure 34. Concluded.

The derivates C_{lp}, C_{np}, C_{mq}, C_{lr}, C_{nr} were calculated using the method from Melin [5] and are shown in Figure 35.

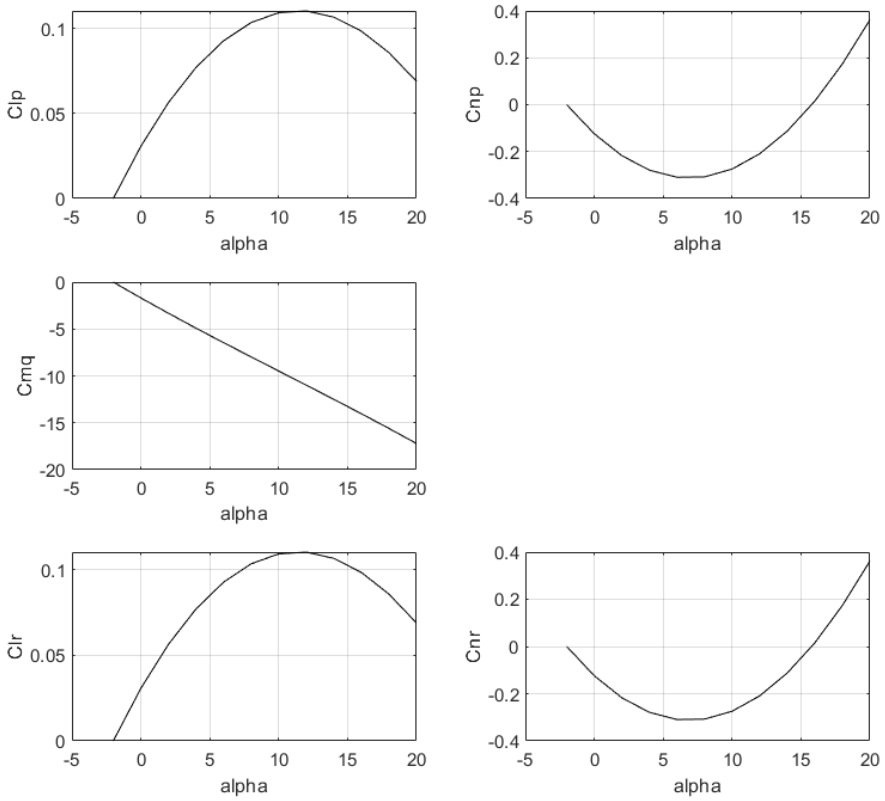


Figure 35. Dynamic derivatives

3.2.4 Static Stability Analysis

Static stability is the initial tendency to return to an equilibrium state after a disturbance [10]. The static stability of the glider was analyzed using traditional methods. Longitudinal stability was determined using Figure 36.

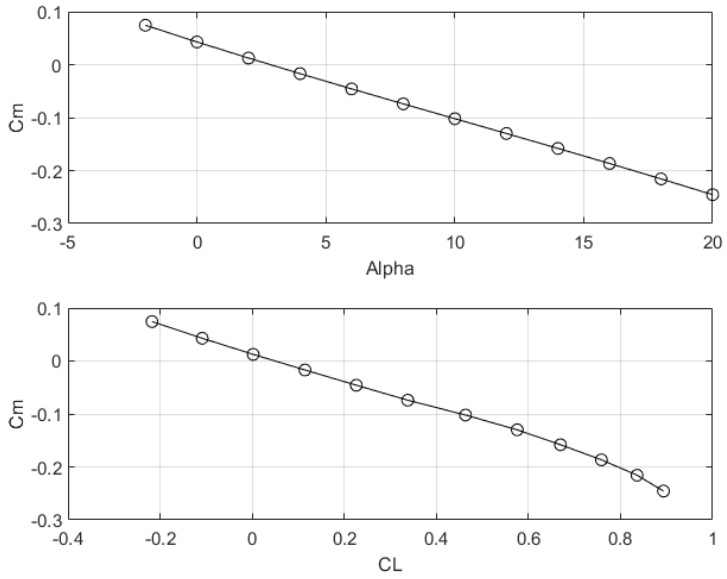


Figure 36. Longitudinal Static Stability

To be statically stable, the pitching moment curve should have a negative slope. Additionally, the pitching moment should have a positive intercept in order to trim at positive angles of attack. A statically stable vehicle would also have a negative slope when plotting C_m against C_L. As shown in Figure 36, the glider meets the static longitudinal stability criteria. Lateral stability was determined using Figure 37.

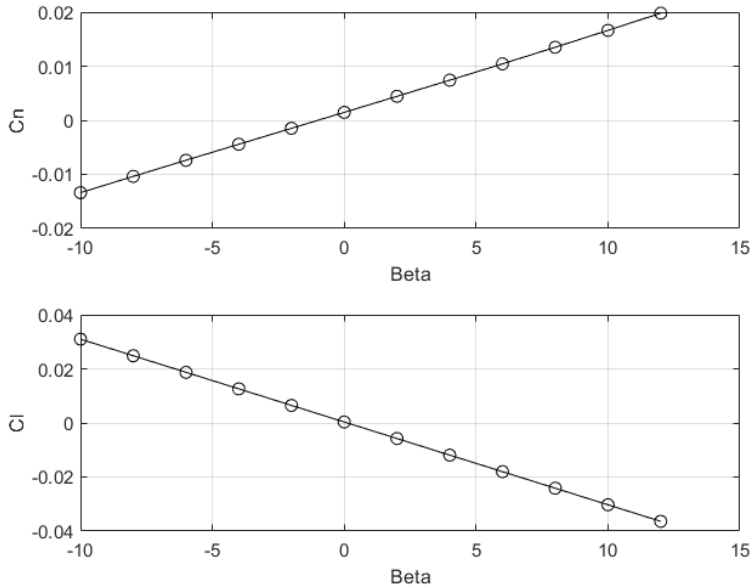


Figure 37. Lateral Static Stability

A positive yawing moment curve indicates the vehicle will return to an equilibrium condition after subjected to a yawing disturbance. A negative rolling moment will restore a vehicle if disturbed from wings-level attitude [10]. Figure 37 describes static lateral stability.

CHAPTER 4. LIFTING BODY GLIDER SIMULINK MODEL

As in Jackson [2], the Simulink model uses the conventional build up method of calculating the aerodynamic coefficients. These incremental coefficients consist of the basic configuration and control surface deflection coefficients. Dynamic derivatives are added to the moment coefficients. The Equations 23 – 28 define now the coefficients are built up to determine the final result of the configuration at any point in time.

$$C_X = C_{X,0}(\alpha,\beta) + C_{X_{\delta_e}}(\alpha)\delta_e + C_{X_{\delta_a}}(\alpha,\beta)\delta_a + C_{X_{\delta_{bf}}}(\alpha)\delta_{bf} \tag{23}$$

$$C_Y = C_{Y_\beta}\beta + C_{Y_{\delta_a}}(\alpha)\delta_a \tag{24}$$

$$C_Z = C_{Z,0}(\alpha,\beta) + C_{Z_{\delta_e}}(\alpha)\delta_e + C_{Z_{\delta_a}}(\alpha,\beta)\delta_a + C_{Z_{\delta_{bf}}}(\alpha)\delta_{bf} \tag{25}$$

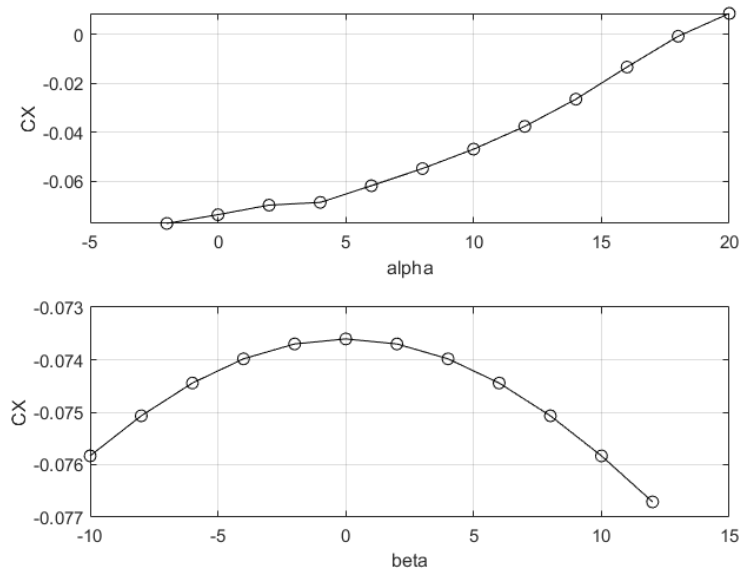
$$C_l = C_{l_\beta}\beta + C_{l_{\delta_a}}(\alpha,\beta)\delta_a + C_{l_p}(\alpha)\frac{pb}{2V} + C_{l_r}(\alpha)\frac{rb}{2V} \tag{26}$$

$$C_m = C_{m,0}(\alpha,\beta) + C_{m_{\delta_e}}(\alpha)\delta_e + C_{m_{\delta_a}}(\alpha,\beta)\delta_a + C_{m_{\delta_{bf}}}(\alpha)\delta_a + C_{m_q}(\alpha)\frac{q\bar{c}}{2V} \tag{27}$$

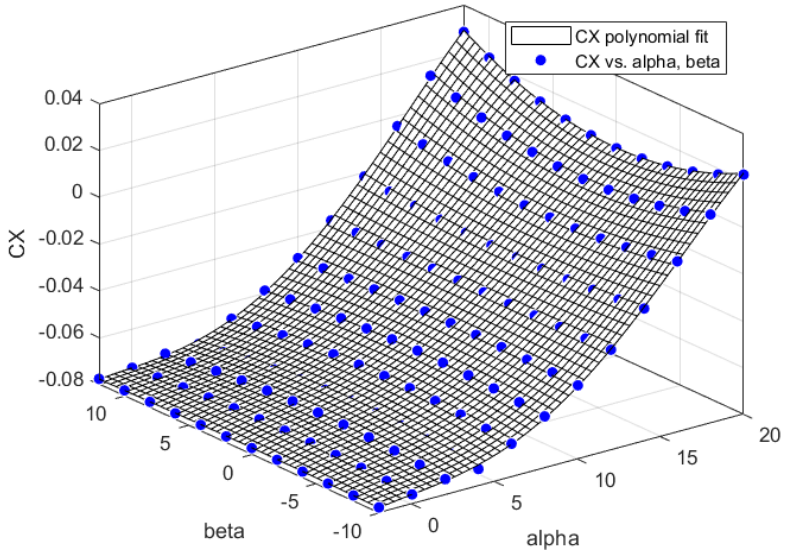
$$C_n = C_{n,0}(\alpha,\beta) + C_{n_{\delta_a}}(\alpha,\beta)\delta_a + C_{n_p}(\alpha)\frac{pb}{2V} + C_{n_r}(\alpha)\frac{rb}{2V} \tag{28}$$

4.1 Preparing the Computational Data

The force and moment coefficients were compiled into polynomial equations as a function of angle of attack and sideslip using the MATLAB Curve Fitting Toolbox. The order of the polynomial curve fit depended on the goodness of fit. Figures 38 through 41 show the coefficient values obtained using the computational model and their respective polynomial function surfaces plotted in three-dimensions.

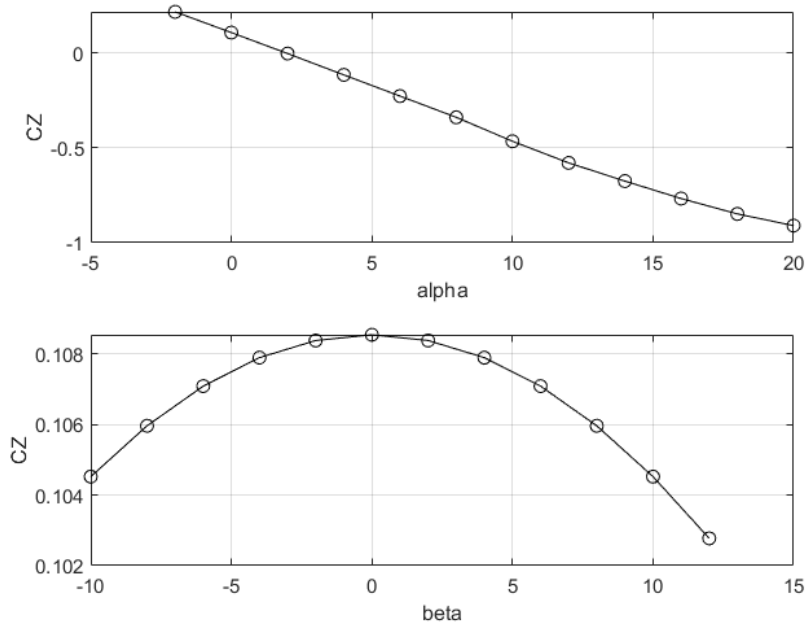


(a) C_X vs. α (deg) at β = 0 and C_X vs. β (deg) at α = 0

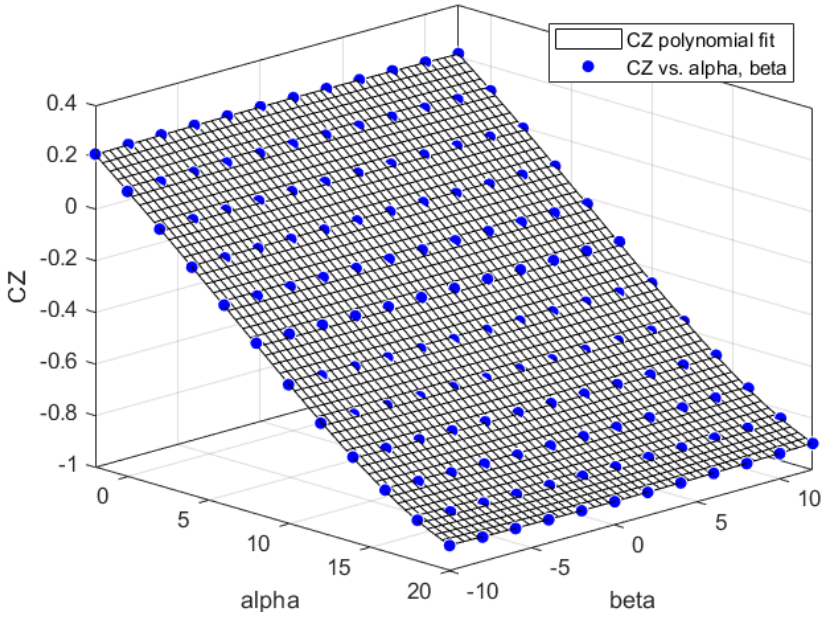


(b) C_X polynomial surface with β (deg) and α (deg)

Figure 38. Computational model data and polynomial curve fit for axial-force coefficient C_X

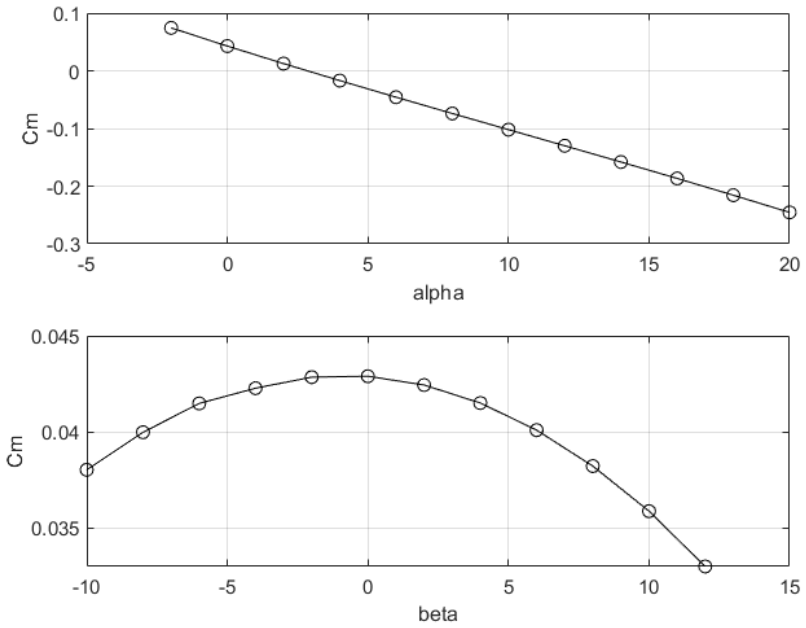


(a) C_Z vs. α (deg) at β = 0 and C_Z vs. β (deg) at α = 0

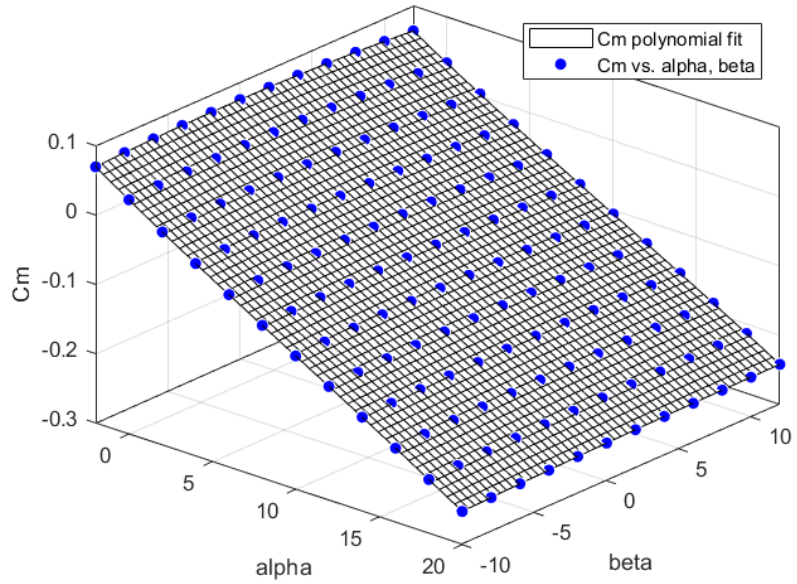


(b) C_Z polynomial surface with β (deg) and α (eg)

Figure 39. Computational model data and polynomial curve fit for normal-force coefficient C_Z

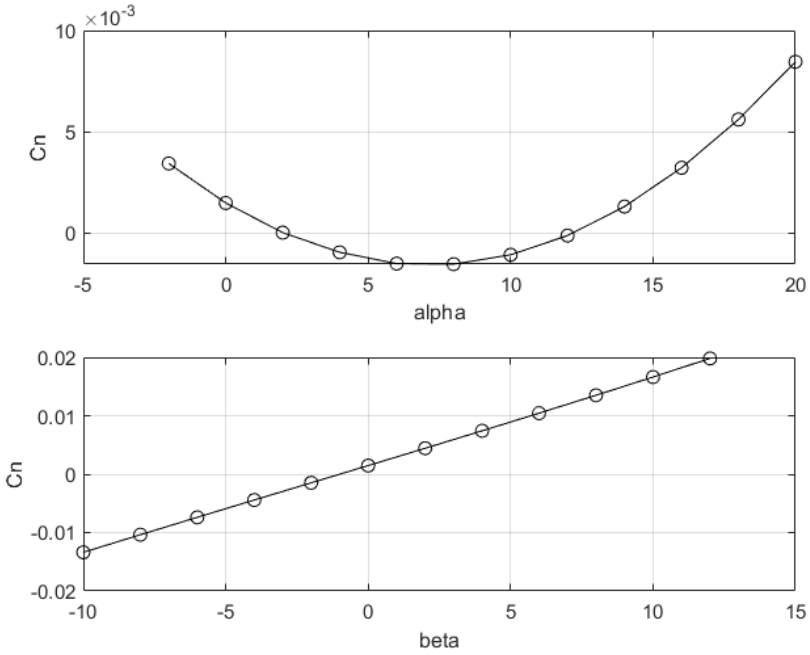


(a) C_m vs. α (deg) at β = 0 and C_m vs. β (deg) at α = 0

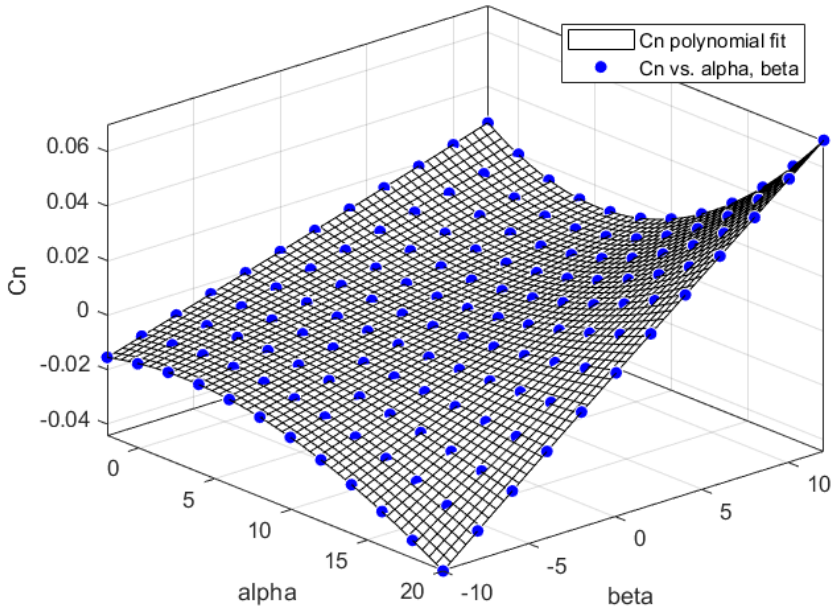


(b) C_m polynomial surface with β (deg) and α (deg)

Figure 40. Computational model data and polynomial curve fit for pitch-moment coefficient C_m



(a) C_n vs. α (deg) at β = 0 and C_n vs. β (deg) at α = 0



(b) C_n polynomial surface with β (deg) and α (deg)

Figure 41. Computational model data and polynomial curve fit for yaw-moment coefficient C_n

The side force coefficient C_Y and rolling moment coefficient C_l, shown in Figures 30 and 31 respectively, were found to be relatively constant with angles of sideslip and attack and were therefore modeled as the following linear equation slopes:

$$C_{Y_\beta} = -0.0052$$

$$C_{l_\beta} = -0.0031$$

As shown by Jackson [2], the polynomial functions are compiled into matrix equations. Figure 42 shows the general matrix equation used to generate the curve fits for the aerodynamic forces C_X and C_Z and the respective polynomial coefficient matrix. Figure 43 shows the matrix equation and polynomial coefficients for the aerodynamic moments C_m and C_n.

$$[C_X \quad C_Z]$$

$$= [1 \quad \beta \quad \alpha \quad \beta^2 \quad \beta\alpha \quad \alpha^2 \quad \beta^3 \quad \beta^2\alpha \quad \beta\alpha^2 \quad \alpha^3 \quad \beta^4 \quad \beta^3\alpha \quad \beta^2\alpha^2 \quad \beta\alpha^3 \quad \alpha^4]P_{force}$$

$$P_{force} = \begin{bmatrix} -0.0735 & 0.1099 \\ 0.0015 & -0.0542 \\ -2.8385 \times 10^{-6} & -1.1477 \times 10^{-6} \\ -5.5346 \times 10^{-5} & -0.0004 \\ -2.0720 \times 10^{-6} & -4.2415 \times 10^{-6} \\ -2.4048 \times 10^{-5} & -4.0494 \times 10^{-5} \\ 2.4241 \times 10^{-5} & 2.5861 \times 10^{-6} \\ 4.0067 \times 10^{-7} & 7.2887 \times 10^{-7} \\ -6.8722 \times 10^{-6} & 8.6972 \times 10^{-6} \\ 3.0895 \times 10^{-8} & 2.7036 \times 10^{-9} \\ -7.4749 \times 10^{-7} & 1.3974 \times 10^{-6} \\ -1.5307 \times 10^{-8} & -2.7011 \times 10^{-8} \\ 7.5201 \times 10^{-7} & -1.1552 \times 10^{-7} \\ -3.4758 \times 10^{-9} & -2.9875 \times 10^{-10} \\ 1.3440 \times 10^{-8} & 2.5136 \times 10^{-9} \end{bmatrix}$$

Figure 42. Matrix equation and coefficient values for aerodynamic forces C_X and C_Z

$$[C_m \quad C_n]$$

$$= [1 \quad \beta \quad \alpha \quad \beta^2 \quad \beta\alpha \quad \alpha^2 \quad \beta^3 \quad \beta^2\alpha \quad \beta\alpha^2 \quad \alpha^3 \quad \beta^4 \quad \beta^3\alpha \quad \beta^2\alpha^2 \quad \beta\alpha^3 \quad \alpha^4]P_{moment}$$

$$P_{moment} = \begin{bmatrix} 0.0428 & 0.0014 \\ -0.0154 & -0.0008 \\ -0.0001 & 0.0014 \\ 0.0001 & 5.7055 \times 10^{-5} \\ 2.8335 \times 10^{-5} & -0.0001 \\ -5.8270 \times 10^{-5} & 1.5270 \times 10^{-6} \\ -4.6754 \times 10^{-6} & 3.3091 \times 10^{-7} \\ -1.0153 \times 10^{-7} & 2.0455 \times 10^{-5} \\ 2.3681 \times 10^{-6} & -2.9173 \times 10^{-7} \\ 3.4580 \times 10^{-9} & 2.0620 \times 10^{-7} \\ 6.3430 \times 10^{-9} & -9.1258 \times 10^{-9} \\ 7.7565 \times 10^{-9} & -1.1752 \times 10^{-7} \\ -1.6603 \times 10^{-8} & -7.9562 \times 10^{-9} \\ -1.2943 \times 10^{-9} & -6.9476 \times 10^{-10} \\ -8.5028 \times 10^{-9} & -1.8019 \times 10^{-10} \end{bmatrix}$$

Figure 43. Matrix equation and coefficient values for aerodynamic moments C_m and C_n

The polynomial functions are used instead of look-up tables to calculate the aerodynamic coefficients in the Simulink model. The matrix equations allow substitute values of sideslip and angle of attack for all possible configurations within the previously mentioned ranges. The resulting matrix is then stored in Simulink and the corresponding coefficient values are calculated by linear interpolation.

4.2 Airframe Model

The airframe is assumed to be rigid and have constant mass, center of gravity, and inertia because the vehicle is an unpowered glider. The airframe model consists of the equations of motion, environmental models, and the calculation of aerodynamic coefficients, forces, and moments.

The aerodynamic data calculated in the MATLAB computational model are used with Simulink blocks to model the airframe. The airframe consists of the Environment, Aerodynamic, and Six Degree-of-Freedom Equations of Motion (6DOF EOM) Models shown in Figure 44.

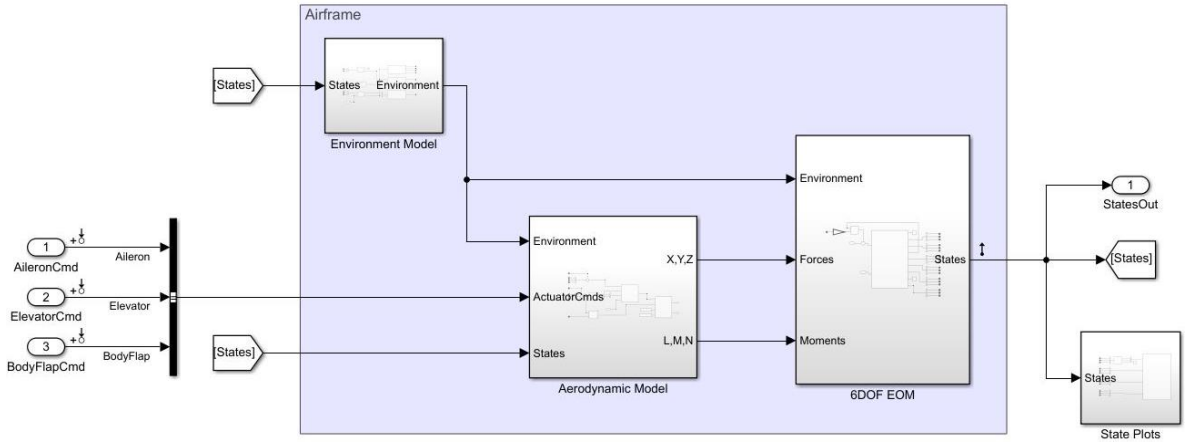


Figure 44. Simulink Airframe Model

The Environment Model subsystem contains the WGS84 Gravity Model, COESA Atmosphere Model block, Dryden Wind Turbulence Model, Incidence & Airspeed, Rotation Angles to Direction Cosine Matrix, and Flat Earth to LLA blocks which are all part of the Aerospace Blockset. The WGS84 Gravity Model block outputs the Earth's gravity at a specific location. The COESA Atmosphere Model block outputs the atmospheric values for temperature, pressure, density, and speed of sound for a given altitude. Wind turbulence is added to the airframe model using the Dryden Wind Turbulence Model (Continuous) block. The Incidence & Airspeed block is used to calculate the airspeed from the x and z velocity components in the body-fixed coordinate frame. The Rotation Angles to Direction Cosine Matrix block determines the direction cosine matrix (DCM) from rotation angles. Geodetic latitude, longitude, and altitude is estimated from the flat Earth position using the Flat Earth to LLA block. The Environment Model subsystem can be seen in Figure 45. The inputs to the Environment Model

are the state attributes of the vehicle, organized in the signal bus State.*variable*, and the outputs are the resulting environmental conditions, organized into the signal bus Environment.*variable*.

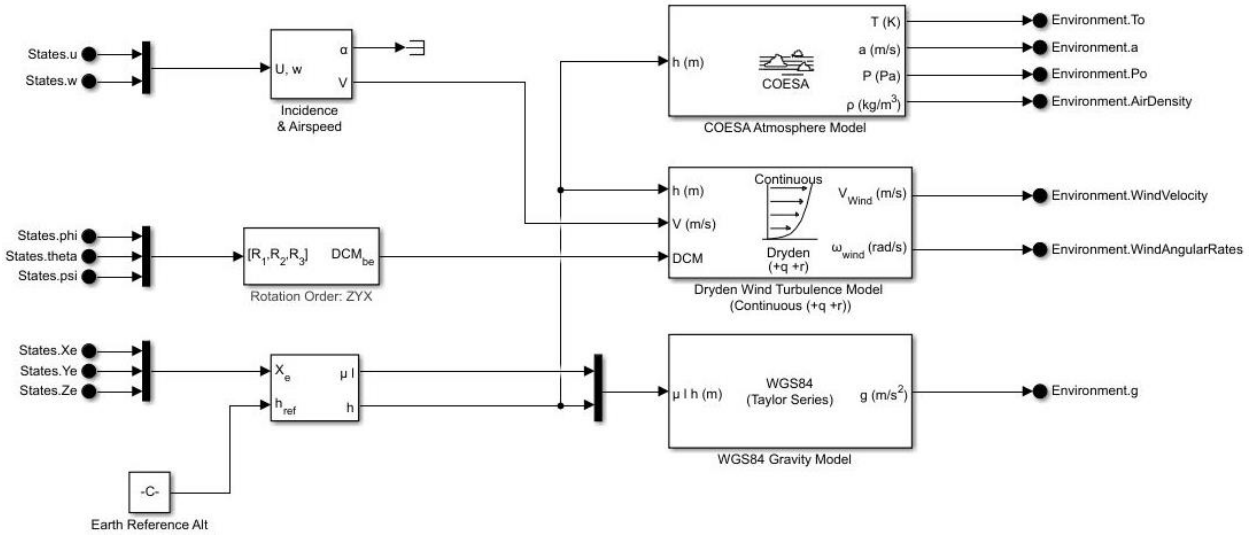


Figure 45. Environment Model subsystem

The environment signal bus is an input to the Aerodynamic Model, shown in Figure 46, in addition to the state signal bus and actuator signals. The Aerodynamic Model subsystem contains the Incidence, Sideslip, & Airspeed, Dynamic Pressure, and Aerodynamic Forces and Moments blocks found in the Aerospace Blockset. The Incidence, Sideslip, & Airspeed block

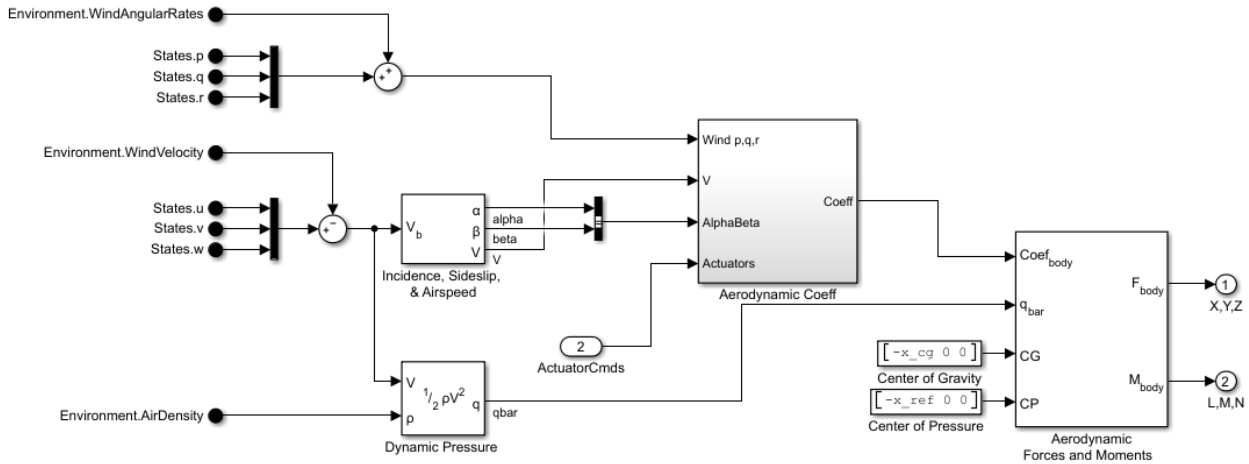


Figure 46. Aerodynamic Model subsystem

calculates incidence, sideslip, and airspeed from velocity components. The calculations conducted in this block are summarized in equations 29 – 31.

$$\alpha = \tan^{-1}\frac{w}{u} \tag{29}$$

$$\beta = \sin^{-1}\frac{v}{V} \tag{30}$$

$$V = \sqrt{u^2 + v^2 + w^2} \tag{31}$$

The Dynamic Pressure block computes the dynamic pressure using equation 32.

$$\bar{q} = \frac{1}{2}\rho V^2 \tag{32}$$

The body forces and moments are calculated in the Aerodynamic Forces and Moments block using the aerodynamic coefficients, dynamic pressure, center of gravity, and center of pressure.

The aerodynamic data are stored in the Aerodynamic Coefficients Subsystem, shown in Figure 47. This subsystem contains the datum, damping derivative, and actuator increment coefficients. The datum coefficients are the aerodynamic coefficients interpolated from polynomial functions determined in the computational model. The datum coefficients block contents are shown in Figure 48.

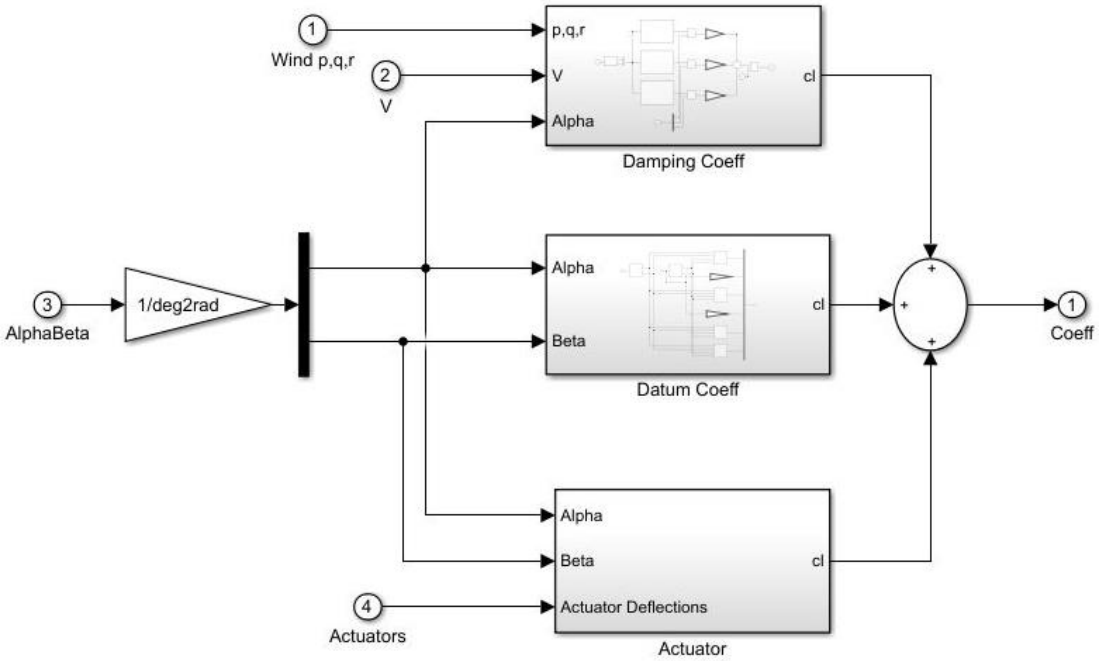


Figure 47. Aerodynamic Coefficients Subsystem

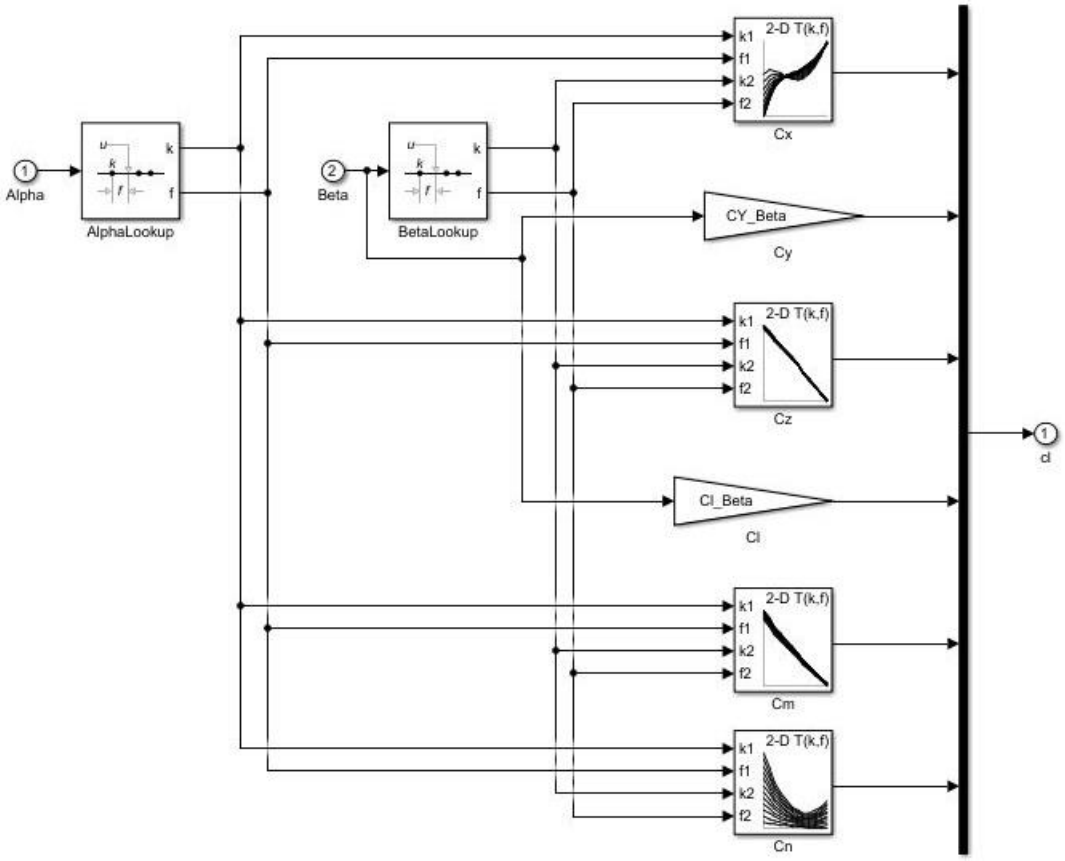


Figure 48. Datum Coefficients Block

The damping coefficients block is used to calculate the dynamic derivatives and modify the moment coefficients to include rotational effects. The contents of the damping coefficient block are shown in Figure 49. The damping derivatives were calculated using the method from Melin's Tornado program [5]. A different boundary condition function was used to incorporate a perturbed value of α, β, p, q, r, and velocity by 0.0001. The MATLAB function can be found in Appendix A. The values used in the Simulink model are p, q, and r, which are calculated from the difference between the datum coefficients and the coefficient results from the respective perturbed p, q, and r boundary conditions.

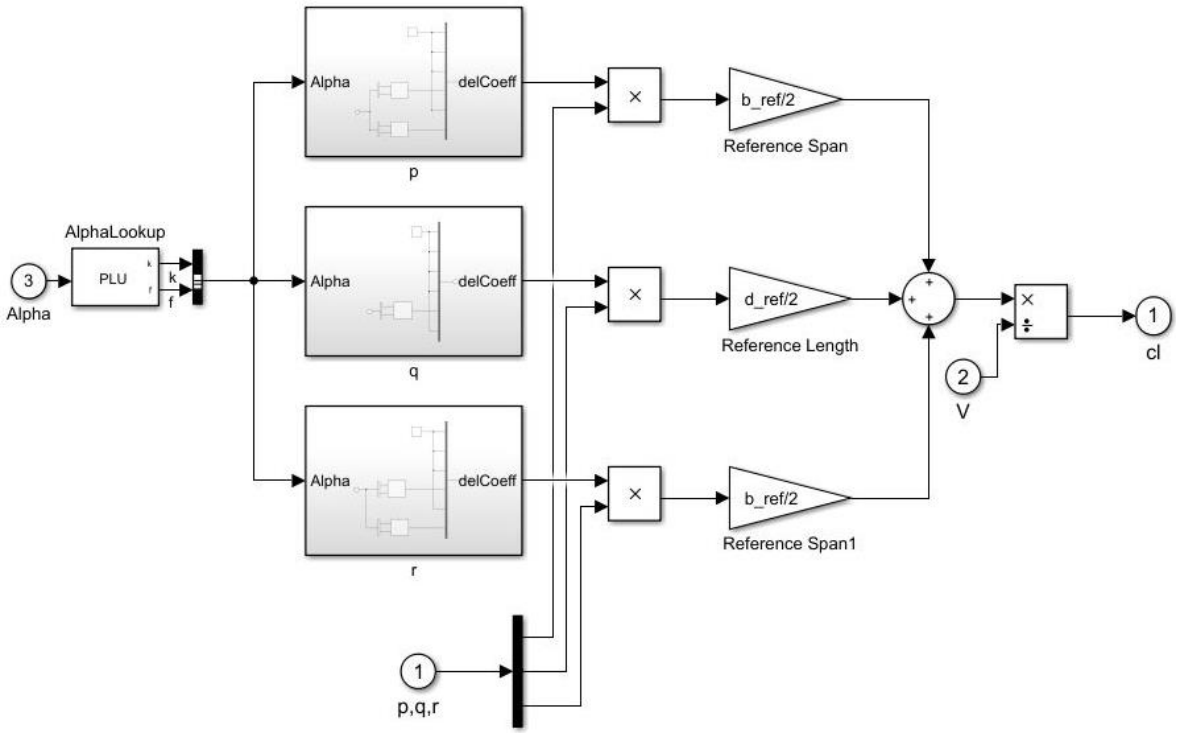


Figure 49. Damping Coefficients Block

The actuator block shown in Figure 50 contains the polynomial functions relative to the contribution of an actuator deflection. To calculate the control surface deflections, the computational model was run with a 2° change from the control surfaces nominal position to calculate symmetric wing flap deflection, i.e., elevator, differential wing flap deflections, i.e., aileron, and body flap deflection. The difference between the results with actuator deflections

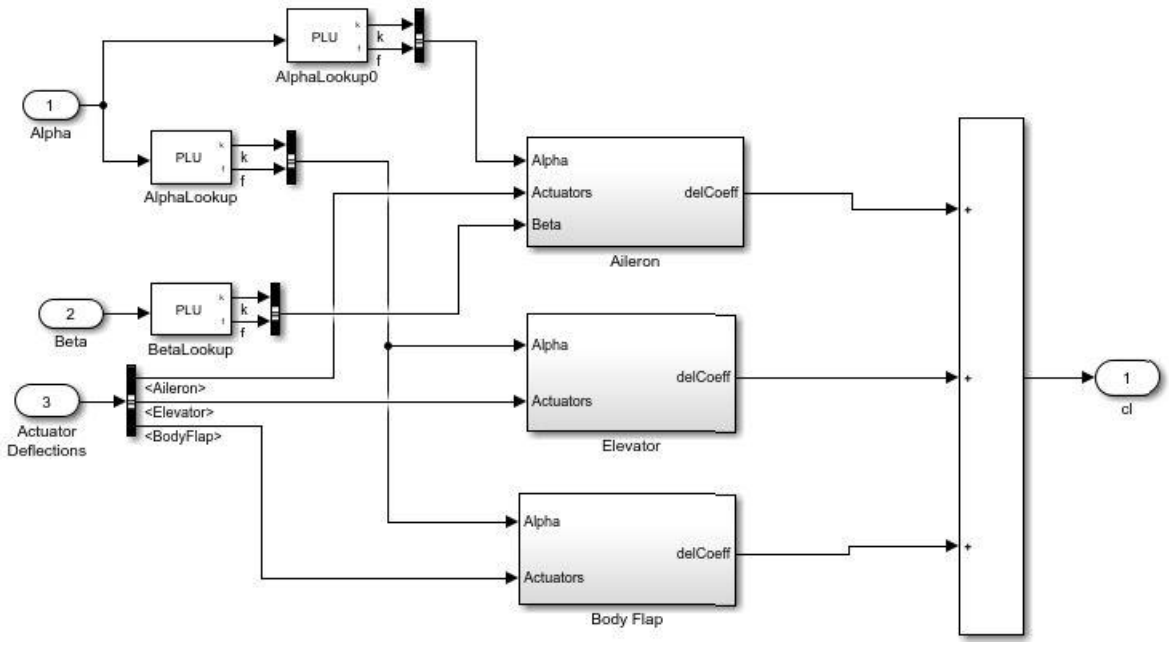


Figure 50. Actuator Block

and the datum were calculated to determine the contribution due to control surface deflection per degree. As an example of how these measurements are incorporated into the Simulink model, Figure 51 shows the elevator contributions. Shown are the longitudinal contributions, for the

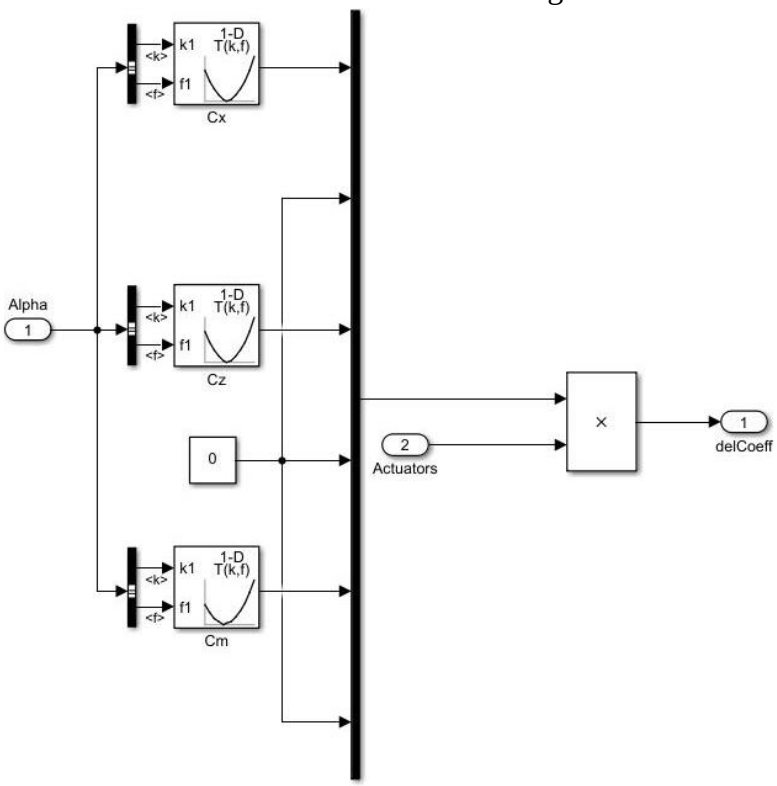


Figure 51. Elevator Block

elevator lateral contributions are negligible, as a function of α. Once a value has been interpolated from the polynomial function, it is multiplied by the amount of deflection, or the number of degrees of deflection.

The 6DOF EOM block, shown in Figure 52, from the Aerospace Blockset implements the Euler angle representation of the equations of motion from the aerodynamic forces and moments. The block assumes a body-fixed coordinate frame where the center of gravity of the

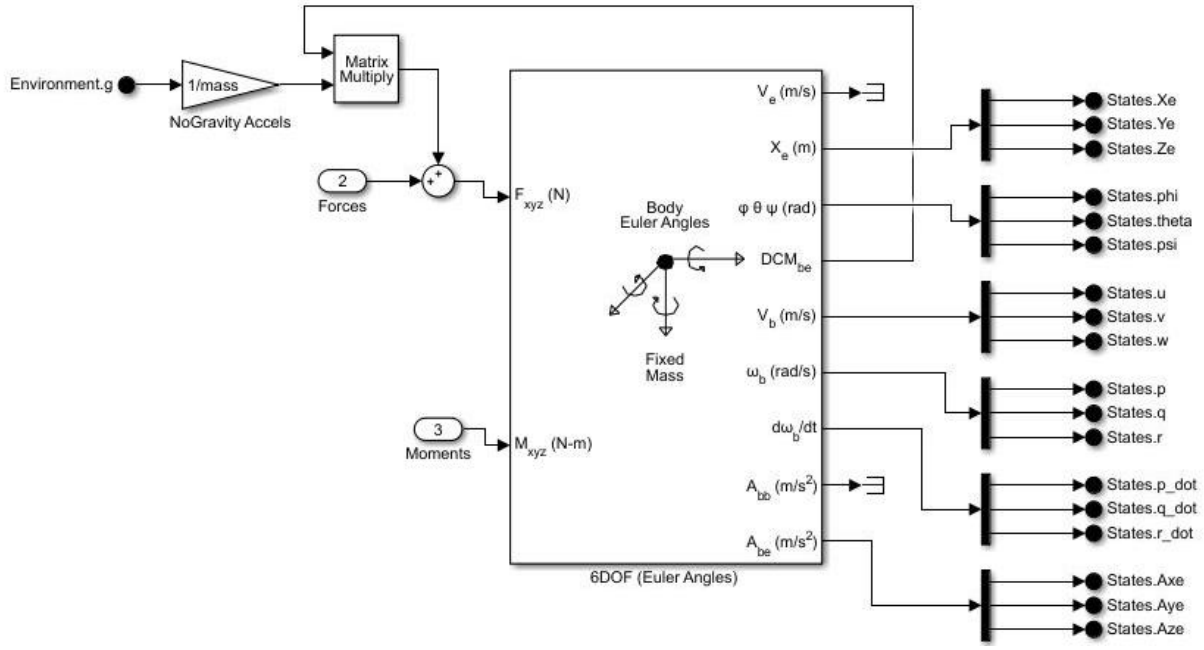


Figure 52. 6DOF EOM Block

body is the origin and the body is assumed to be rigid. The equations used in this block to determine the output state values are shown in equations 33 – 40.

$$\bar{F}_b = \begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} = m\left(\dot{\bar{V}}_b + \bar{\omega} \times \bar{V}_b\right) \tag{33}$$

$$A_{bb} = \begin{bmatrix} \dot{u}_b \\ \dot{v}_b \\ \dot{w}_b \end{bmatrix} = \frac{1}{m}\bar{F}_b - \bar{\omega} \times \bar{V}_b \tag{34}$$

$$A_{be} = \frac{1}{m}F_b \tag{35}$$

$$\bar{V}_b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \bar{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{36}$$

$$\bar{M}_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = I\dot{\bar{\omega}} + \bar{\omega} \times (I\bar{\omega}) \tag{37}$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{38}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \equiv J^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{39}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & \frac{sin\phi}{cos\theta} & \frac{cos\phi}{cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{40}$$

Here, the forces are in the body fixed frame and the mass is assumed constant. A_{bb} is the acceleration of the body with respect to the body reference frame and A_{bi} is with respect to the inertial frame. This block considers the flat earth reference frame to be inertial. The inertial tensor, I, is with respect to the origin at the center of gravity. The relationship between the angular velocity and rate of change of the Euler angles is used to determine the Euler rates. These equations were obtained from MathWorks [11].

4.3 Dynamic Stability Analysis

A dynamic stability analysis was conducted using eigenvalue analysis to determine the longitudinal and lateral flying qualities using the airframe modeled in Simulink. MATLAB functions were used to linearize and trim the dynamic system modeled in Simulink. Simulink

Control Design was used to generate a trimmed operating point and derive a linear state-space model. The state-space model is used to compute the longitudinal and lateral flight modes used to determine stability.

First, a trim condition must be defined as the operating point which the model is to be linearized around. The component of the position X_e is set to non-steady state and non-zero initial conditions are defined. The output of the MATLAB operspec function used to set the initial conditions is shown in Table 2, with the Dryden Wind Turbulence Model states omitted. The minimum and maximum control surface deflection values are also specified to be +/– 20°.

States:	Inputs:	Outputs:
(1.) phi spec: dx = 0, initial guess: 0	(1.) GliderFlightAnalysis/AileronCmd initial guess: 0	spec: none
(2.) theta spec: dx = 0, initial guess: -0.34	(2.) GliderFlightAnalysis/ElevatorCmd initial guess: 0	spec: none
(3.) psi spec: dx = 0, initial guess: 0	(3.) GliderFlightAnalysis/BodyFlapCmd initial guess: 0	spec: none
(4.) p spec: dx = 0, initial guess: 0		spec: none
(5.) q spec: dx = 0, initial guess: 0		spec: none
(6.) r spec: dx = 0, initial guess: 0		spec: none
(7.) Ubody spec: dx = 0, initial guess: 15		spec: none
(8.) Vbody spec: dx = 0, initial guess: 0		spec: none
(9.) Wbody spec: dx = 0, initial guess: -0.402		spec: none
(10.) Xe spec: dx = 0, initial guess: 0		spec: none
(11.) Ye spec: dx = 0, initial guess: 0		spec: none
(12.) Ze spec: dx = 0, initial guess: -127		spec: none

Table 2. Operating point specification for the Model GliderFlightAnalysis.

The MATLAB function findop() is used to find an operating point that satisfies the defined trim condition constraints. An Operating Point Search Report is generated from the TrimAirframe() function and dictates whether the specifications defined were successfully met. The Operating Point Search Report can be found in Appendix B.

Prior to linearizing the system, a set of linear analysis points are defined. The linio function is used to create a linear analysis point for the Simulink model for each of the input actuator commands and output states. In the Simulink model, perturbations are applied to the actuator input commands and measurements are taken for the output states. Setlinio is then used to save the linear analysis points to the Simulink model. The linearize function is used to linearize the model around the trimmed flight condition. The full linear system model from MATLAB is shown in Figure 53. The full linear system model is decoupled into reduced order models for the longitudinal and lateral axes. The longitudinal and lateral state and control vectors are shown in Equations 41 and 42, respectively.

$$\mathbf{x} = [u \quad w \quad q \quad \theta]^T \tag{41}$$

$$\eta = [\delta_e \quad \delta_{bf}]$$

$$\mathbf{x} = [v \quad p \quad r \quad \phi]^T \tag{42}$$

$$\eta = [\delta_a]$$

A =

	phi	theta	psi	p	q	r	Ubody	Vbody	Wbody
phi	-5.53e-25	-4.085e-24	0	1	8.586e-07	-0.001341	0	0	0
theta	4.085e-24	0	0	0	1	0.0006404	0	0	0
psi	4.125e-22	5.477e-27	0	0	-0.0006404	1	0	0	0
p	-2.169	1.543	-1.05	0.006407	0	0.06961	-0.02374	-0.6896	0.03824
q	3.302	0.8378	3.043	-3.345e-24	-0.5145	-1.887e-22	0.1618	0.008948	-0.6904
r	0.2925	-0.3332	0.05132	-0.0133	1.887e-22	-0.01072	0.007977	0.1006	0.01895
Ubody	-0.07425	-0.3196	-0.1032	0	0.01152	-4.124	-0.01342	0.004758	0.02675
Vbody	0.1866	0.0529	-0.03903	-0.01152	0	-6.622	3.138e-05	-0.027	-5.41e-06
Wbody	0.6196	1.114	1.385	4.124	6.622	0	-0.02194	0.006898	-0.3341
Xe	-1.689e-11	-5.018e-11	3.441e-11	0	0	0	0.8488	-0.5287	-0.001477
Ye	0.01046	-5.939e-12	7.801	0	0	0	0.5287	0.8488	-0.0001653
Ze	-4.124	-6.622	0	0	0	0	0.001341	-0.0006404	1

	Xe	Ye	Ze
phi	0	0	0
theta	0	0	0
psi	0	0	0
p	0	0	0.003308
q	0	0	0.001693
r	0	0	0.0003748
Ubody	0	0	4.66e-05
Vbody	0	0	-0.0001302
Wbody	0	0	0.001175
Xe	0	0	0
Ye	0	0	0
Ze	0	0	0

Figure 53. Full linear system model

B =

	Aileron	Elevator	BodyFlap
phi	0	0	0
theta	0	0	0
psi	0	0	0
p	-0.01862	0	0
q	-0.004762	-0.01052	-0.02371
r	0.002345	0	0
Ubody	-5.551e-05	9.882e-05	3.976e-05
Vbody	-0.000272	0	0
Wbody	-0.0004887	-0.00104	-0.001222
Xe	0	0	0
Ye	0	0	0
Ze	0	0	0

C =

	phi	theta	psi	p	q	r	Ubody	Vbody	Wbody	Xe	Ye	Ze
phi	1	0	0	0	0	0	0	0	0	0	0	0
theta	0	1	0	0	0	0	0	0	0	0	0	0
psi	0	0	1	0	0	0	0	0	0	0	0	0
p	0	0	0	1	0	0	0	0	0	0	0	0
q	0	0	0	0	1	0	0	0	0	0	0	0
r	0	0	0	0	0	1	0	0	0	0	0	0
Ubody	0	0	0	0	0	0	1	0	0	0	0	0
Vbody	0	0	0	0	0	0	0	1	0	0	0	0
Wbody	0	0	0	0	0	0	0	0	1	0	0	0
Xe	0	0	0	0	0	0	0	0	0	1	0	0
Ye	0	0	0	0	0	0	0	0	0	0	1	0
Ze	0	0	0	0	0	0	0	0	0	0	0	1

D =

	Aileron	Elevator	BodyFlap
phi	0	0	0
theta	0	0	0
psi	0	0	0
p	0	0	0
q	0	0	0
r	0	0	0
Ubody	0	0	0
Vbody	0	0	0
Wbody	0	0	0
Xe	0	0	0
Ye	0	0	0
Ze	0	0	0

Continuous-time state-space model.

Figure 53. Full linear system model (concluded)

4.3.1 Longitudinal Analysis

Transfer functions are generated from the longitudinal model for the relevant input and outputs. Eigenvalue analysis is used to determine the roots of the transfer functions, where it is required that any real root must be negative for the dynamic system to be stable. Longitudinal transfer functions are generated from the input elevator to output states q and θ and are shown in Equations 43 and 44, respectively. The transfer functions response to a step input are shown in Figure 54.

$$q: \frac{-0.01052\,s^3 - 0.002922\,s^2 - 4.138 \times 10^{-5}\,\mathrm{s} - 2.828 \times 10^{-22}}{s^4 + 0.8621\,s^3 + 3.916\,s^2 + 0.5643\,s + 0.02337} \tag{43}$$

$$\theta: \frac{-0.01052\,s^2 - 0.002922\,\mathrm{s} - 4.138 \times 10^{-5}}{s^4 + 0.8621\,s^3 + 3.916\,s^2 + 0.5643\,s + 0.02337} \tag{44}$$

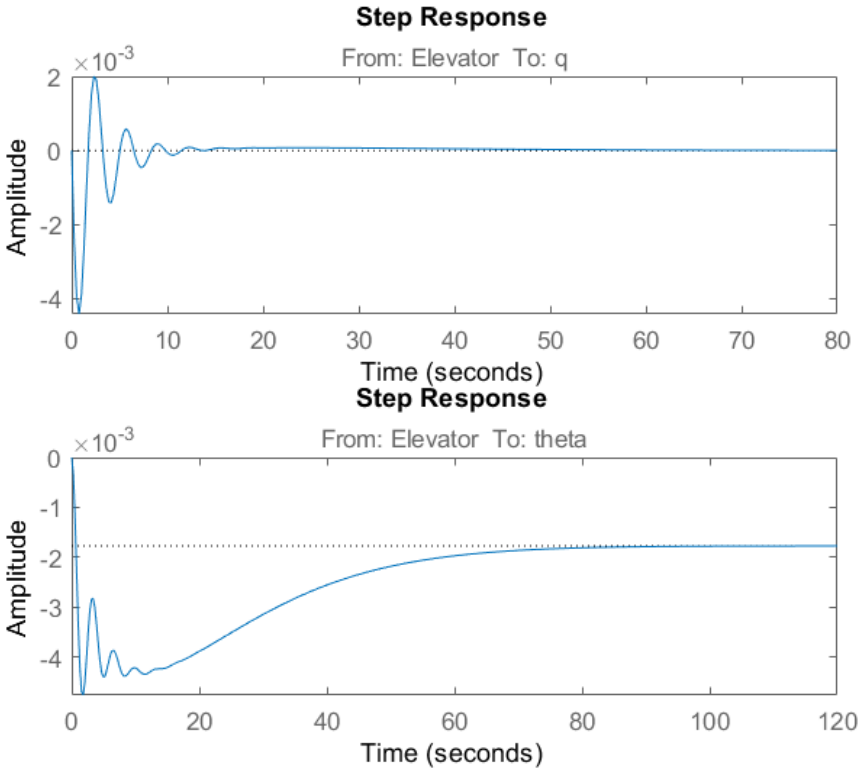


Figure 54. Longitudinal transfer functions response to step input

The MATLAB function eig is used to solve for the roots of the state-space longitudinal model, which are shown in Equation 45. A pole-zero plot is used to visualize the roots, shown in Figure 55. As shown, the real roots of the state-space longitudinal model are negative indicating the glider is longitudinally stable.

$$\lambda_{long} = \begin{matrix} -0.3574 + 1.9174\mathrm{i} \\ -0.3574 - 1.9174\mathrm{i} \\ -0.0736 + 0.0270\mathrm{i} \\ -0.0736 - 0.0270\mathrm{i} \end{matrix} \tag{45}$$

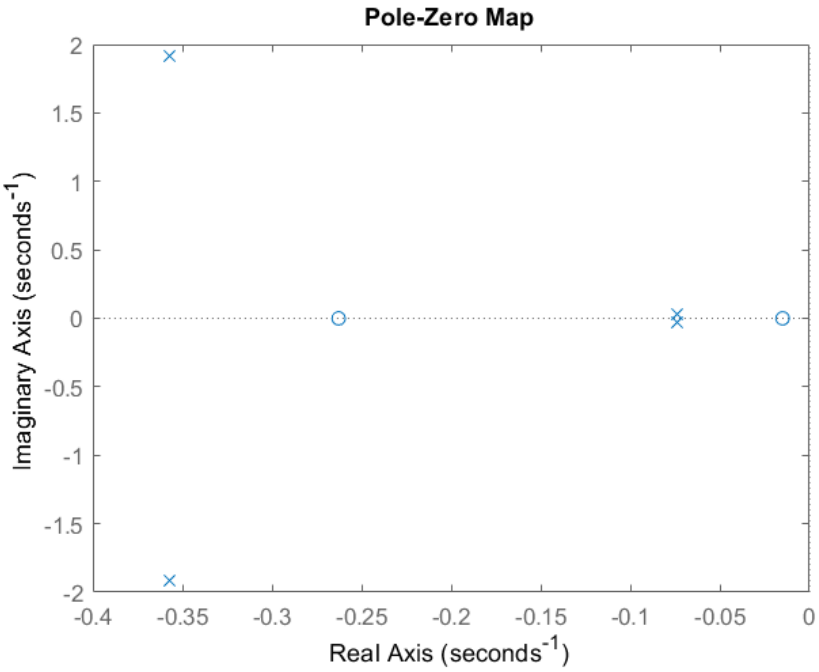


Figure 55. Longitudinal Pole-Zero plot

Longitudinal perturbations on an aircraft typically result in two modes of under damped oscillatory motion. The short-period mode has a short period and is heavily damped. The long-period, or phugoid, mode has a much longer period and is more lightly damped [13]. Inherently stable aircraft tend to have complex conjugate roots. The imaginary part of the roots is an indicator of the frequency of oscillation. The greater of the imaginary root is the short-period mode while the lower imaginary root is the long-period, or phugoid mode. The longitudinal damping ratio, ζ, and undamped natural frequency, ω_n, of each mode can be computed using the relationships in Equations 46 and 47 [12].

$$\sigma_{sp} = \zeta_{sp} * \omega_{n_{sp}} \pm i * \omega_{n_{sp}} * \sqrt{1 - \zeta_{sp}^2} \tag{46}$$

$$\sigma_{ph} = \zeta_{ph} * \omega_{n_{ph}} \pm i * \omega_{n_{ph}} * \sqrt{1 - \zeta_{ph}^2} \tag{47}$$

Using the computed linear roots, the short-period characteristics are found to be:

$$\zeta_{sp} = 0.1833$$

$$\omega_{sp} = 1.9505$$

and the long-period, or phugoid, characteristics:

$$\zeta_{ph} = 0.9388$$

$$\omega_{ph} = 0.0784$$

Flying qualities are related to the dynamic and control characteristics of the airplane. The short- and long-period damping ratios and undamped natural frequencies are used to determine how easy or difficult the airplane is to fly [10]. The MIL-F-8785C standard for longitudinal flying qualities for long-period oscillations (phugoid) that occur when an aircraft seeks a stabilized airspeed following a disturbance is that the oscillations must meet the following minimum requirements:

Level 1: $\zeta_{ph} >= 0.04$

Level 2: $\zeta_{ph} >= 0.0$

Level 3: $T_{2_{ph}} >= 55\ s$

where $T_{2_{ph}}$ is the time-to-double phugoid amplitude [12]. Levels 1-3 correspond to Cooper-Harper Scale values 1-3.5, 3.5-6.5, and 6.5-9+. Time-to-double amplitude is calculated using Equation 48.

$$T_{2_{ph}} = \frac{ln(2)}{-\zeta_{_{ph}} * \omega_{_{ph}}}\ s \tag{48}$$

$$T_{2_{ph}} = -9.4193\ s$$

Here, the time-to-double amplitude is negative which indicates the value is represents the time-to-halve the phugoid amplitude, or $T_{1/2_{ph}}$. Thus, with $\zeta_{ph} = 0.9388$, the glider meets the Level 1 criteria of the MIL-F-8785C standard.

The short-period-mode flying qualities are determined by upper and lower limits for the short-period damping ratio, per the MIL-F-8785C standard. Low damping can cause difficult short-period oscillatory response, while high damping can cause slowed response to control inputs [12]. The short-period damping ratio limits are:

Level 1: $0.30 < \zeta_{sp} < 2.00$

Level 2: $0.20 < \zeta_{sp} < 2.00$

Level 3: $0.15 < \zeta_{sp}$

With $\zeta_{sp} = 0.1833$, the glider meets the Level 3 criteria of the MIL-F-8785C standard.

4.3.2 Lateral Analysis

Transfer functions are generated from the lateral model for the relevant input and outputs. Eigenvalue analysis is used to determine the roots of the transfer functions, where it is required that any real root must be negative for the dynamic system to be stable. Lateral transfer functions are generated from the input aileron to output states p and ϕ and are shown in Equations 49 and 50, respectively. The transfer functions response to a step input are shown in Figure 56.

$$p: \frac{-0.01862\,s^3 - 0.0003516\,s^2 - 0.001692\,s - 8.292 \times 10^{-8}}{s^4 + 0.03132\,s^3 + 2.829\,s^2 + 0.2467\,s + 0.1089} \tag{49}$$

$$\phi: \frac{-0.01862\,s^2 - 0.0003519\,s - 0.001692}{s^4 + 0.03132\,s^3 + 2.829\,s^2 + 0.2467\,s + 0.1089} \tag{50}$$

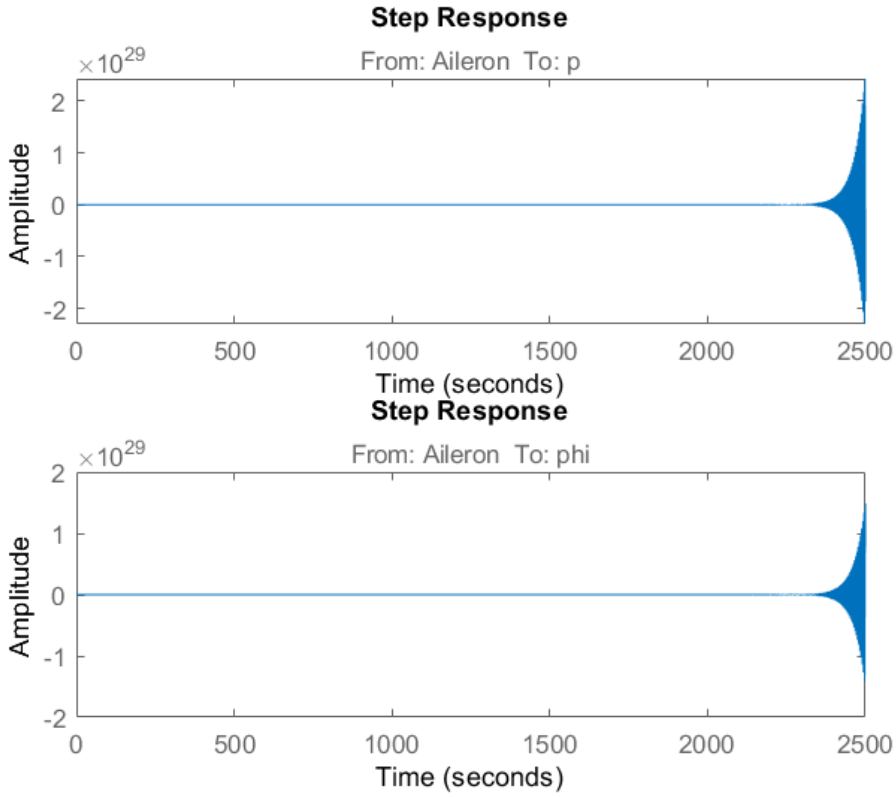


Figure 56. Lateral transfer functions response to step input

The MATLAB function eig is used to solve for the roots of the state-space lateral model, which are shown in Equation 51. A pole-zero plot is used to visualize the roots, shown in Figure 57. As shown, the real roots of the state-space lateral model not typical results for stable aircraft. A laterally stable vehicle will have one complex root and two real roots, indicating the dutch roll, spiral, and roll modes. The glider has two complex roots, one positive and one negative, indicating the glider is not laterally stable. This is likely be due to the absence of a rudder, large side force on the fuselage, or low lateral damping.

$$\lambda_{lat} = \begin{matrix} 0.0289 \ + \ 1.6715\text{i} \\ 0.0289 \ - \ 1.6715\text{i} \\ -0.0445 \ + \ 0.1923\text{i} \\ -0.0445 \ - \ 0.1923\text{i} \end{matrix} \tag{51}$$

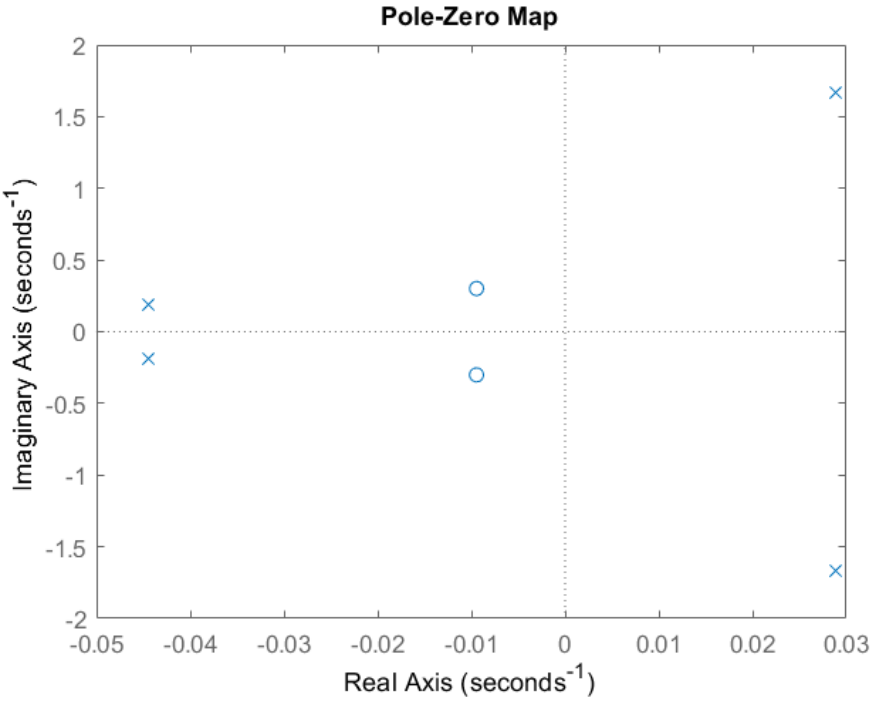


Figure 57. Longitudinal Pole-Zero plot

4.4 Flight Simulation

FlightGear is an open source flight simulator which can be fed external flight dynamics data [13]. The data computed in the Simulink model are exported to FlightGear where the user can visualize the vehicle's flight path and vehicle dynamics. This is helpful to understand the control effectiveness and stability of the model. It is simple enough to set the computational data as the flight dynamics source by using the FlightGear Simulink interface.

4.4.1 FlightGear Aircraft Model Requirements

Within the FlightGear program files, aircraft models must be defined in the *FlightGearRoot*/data/Aircraft/ folder and subfolders. Aircraft data are specified in \FlightGear\data\Aircraft\model folders. The aircraft model must be linked through a master file

named *model-set.xml* in the model subfolder. Simulink can be set as the flight dynamics model source using the command <flight-model>network</flight-model>, as shown in Figure 58.

```
<PropertyList>

 <sim>

  <description>High Altitude Glider</description>
  <author>N. Valentour</author>

  <flight-model>network</flight-model>

  <model>
   <path>Aircraft/glider/Models/glider.xml</path>
  </model>

  <systems>
   <electrical>
    <path>Aircraft/Generic/generic-electrical.xml</path>
   </electrical>
  </systems>

  <help>
   <title>High Altitude Glider</title>
  </help>

 </sim>

</PropertyList>
```

Figure 58. Glider model file in FlightGear

The aircraft data stored in the model folder along with the 3D geometry model file. FlightGear require the 3D geometry to be stored in the aircraft folder in AC3D format. AC3D is not an open source program, so Blender was used to define the 3D geometry and export the file into an AC3D file format [14]. The model could then be viewed in AC3D to determine the locations of the geometry objects. To use a 3D model with FlightGear, each movable surface must be defined as an object within the 3D model. The 3D model was imported into Blender from SolidWorks, with each control surface already a distinct object within the file. In Blender, the objects were assigned names: UpperFlap, LowerFlap, RightAileron, and LeftAileron.

FlightGear requires the hinge lines be defined with respect to the object center and axis of rotation, which is done in AC3D. The coordinate system in FlightGear is different from that used

in AC3D. The FlightGear coordinate system forms a right-handed system which is rotated from the standard aerospace coordinate system by -180° about the y-axis. The AC3D coordinate system is formed by inverting the standard body coordinate axes. A comparison of the two coordinate systems is shown in Figure 59 [16].

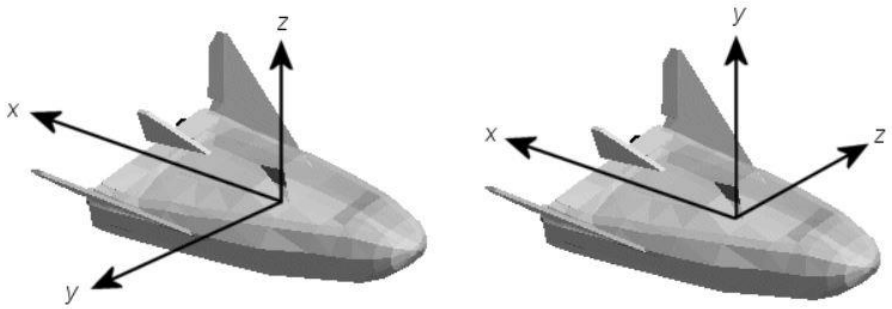


Figure 59. FlightGear Coordinate System vs AC3D Coordinate System

The hinge line of the control surface objects can be determined by selecting two vertices that lie on the hinge line and computing the difference in the x, y, and x directions, this is the relative motion vector in the animation axis. The center of the hinge line is calculated by finding the midpoint between the two vertices in the x, y, and z directions. These steps are combined to obtain the hinge line animation used in FlightGear [17]. An example of the model file is shown in Figure 60.

```
<animation>
 <type>rotate</type>
 <object-name>LeftAileron</object-name>
 <property>/surface-positions/left-aileron-pos-norm</property>
 <factor>30</factor>
 <offset-deg>0</offset-deg>
 <center>
  <x-m>0.9610</x-m>
  <y-m>-0.4770</y-m>
  <z-m>0.0170</z-m>
 </center>
 <axis>
  <x>0.058</x>
  <y>-0.2640</y>
  <z>0.084</z>
 </axis>
</animation>
```

Figure 60. Hinge line animation for the left aileron

4.4.2 Running FlightGear with Simulink Models

The Aerospace Blockset in Simulink contains preconfigured blocks that are used with the flight simulator interface. Separate Simulink models were created for this purpose, using the same airframe subsystem that was previously described. An overview of the simulation model is shown in Figure 61. In this model, a joystick is used to fly the glider within the FlightGear

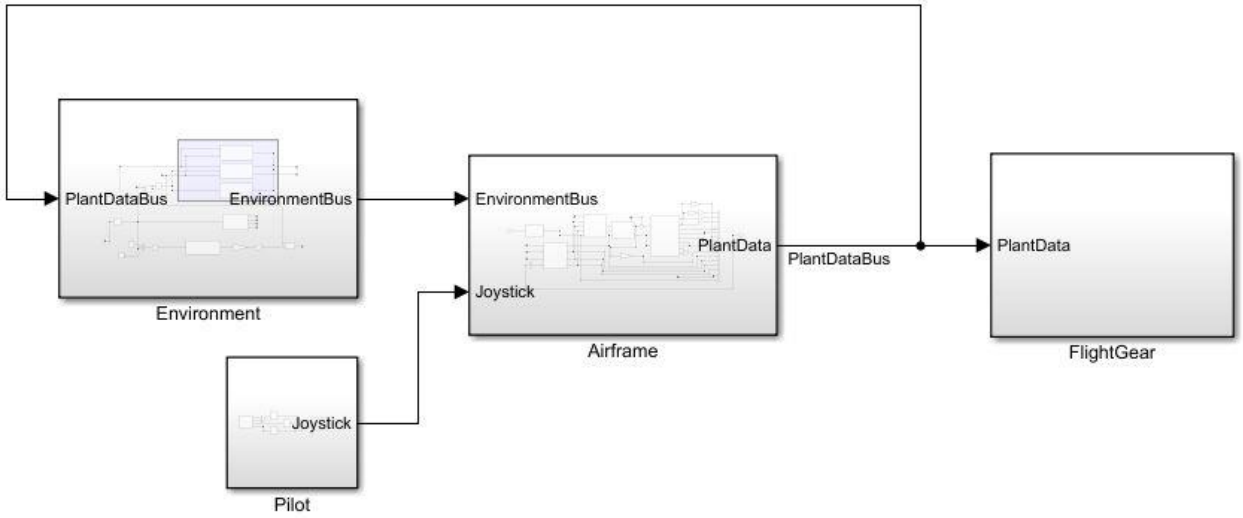


Figure 61. Lifting Body Glider Joystick Model

software. The Pilot Joystick block provides a joystick interface within the Simulink environment. The joystick maps roll, pitch, yaw, and throttle to channels X, Y, R, and Z, respectively. The Pilot subsystem is shown in Figure 62.

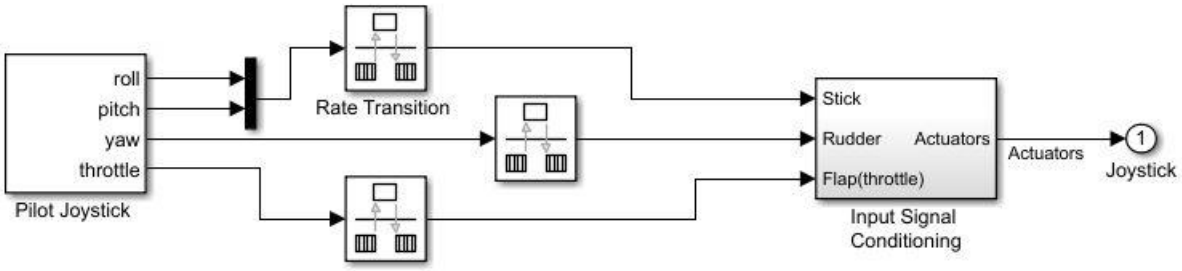


Figure 62. Pilot subsystem

The FlightGear subsystem contains the FlightGear Preconfigured 6DoF Animation block which connects the Simulink model to the FlightGear flight simulator. This block drives attitude

and position values to FlightGear when given longitude (l), latitude (μ), altitude (h), roll (ϕ), pitch (θ), and yaw (Ψ). The FlightGear subsystem is shown in Figure 63.

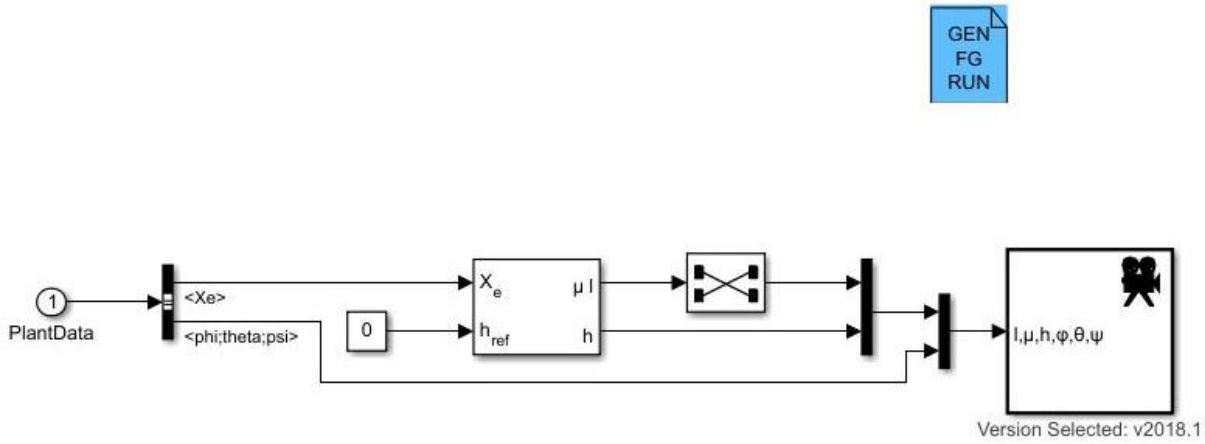


Figure 63. FlightGear subsystem

FlightGear is initiated from MATLAB/Simulink using the Generate Run Script block. The block requests inputs for the FlightGear geometry model name, Airport ID, Runway ID, initial altitude, initial heading, offset distance, and offset azimuth values. Once completed, the Generate Script button will create a run script and save it to the MATLAB working folder. The run script is executed by the command `dos('runfg &')`. The run script for the glider is shown in Figure 64, modified to include a -90° pitch initial condition. A snapshot of the glider during the FlightGear simulation is shown in Figure 65.

```
C:
cd C:\Program Files\FlightGear

SET FG_ROOT=C:\Program Files\FlightGear\data
.\\bin\fgfs --aircraft=Glider --fdm=network,localhost,5501,5502,5503 -
-fog-fastest --disable-clouds --start-date-lat=2004:06:01:09:00:00 --
disable-sound --in-air --enable-freeze --airport=KSFO --runway=10L --
altitude=400 --heading=113 --pitch=-90 --offset-distance=0.5 --offset-
azimuth=0 --enable-terrasync --prop:/sim/rendering/shaders/quality-
level=0
```

Figure 64. FlightGear run script

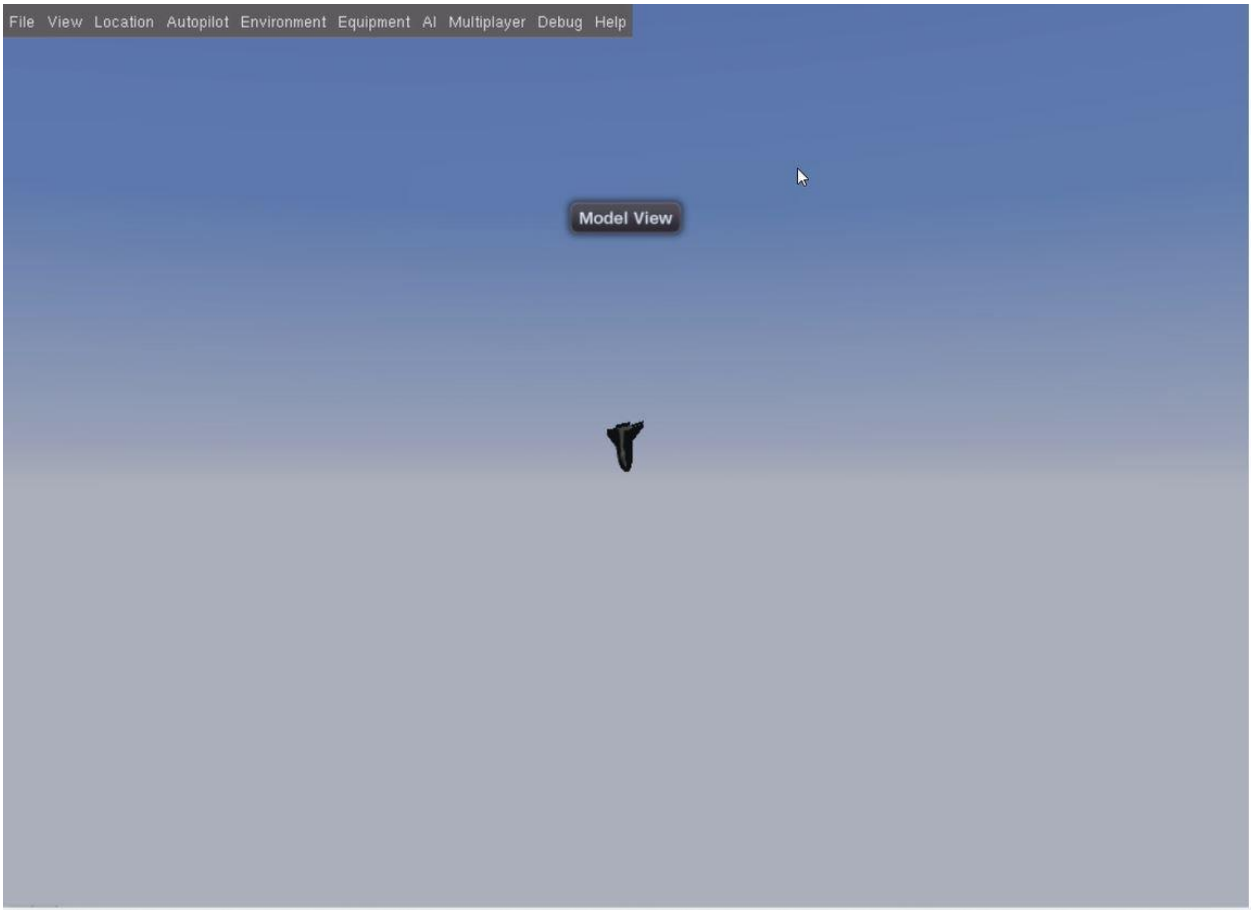


Figure 65. FlightGear simulation snapshot

CHAPTER 5. STRUCTURAL DESIGN

Upon the initial stability results, 3D models were prepared in Solidworks to be submitted for fabrication at the Kennedy Space Center Foam Operations facility. The fabrication models were sections of the glider that were to be cut from high density foam and used to make rigid molds from fiberglass and epoxy resin. The glider was segmented by the body, wings, and control surfaces, with each segment having a top and bottom section. The foam molds are used to fabricate rigid molds from fiberglass and epoxy resin. The anticipated loads calculated in MATLAB and a safety factor of three was used to size structural components and materials to develop the structural design of the vehicle. The load on the ailerons was used to design hinge blocks which will be 3D printed to properly form to the wing curvature. The body flap hinge was designed using the same method, and all control surfaces are designed to deflect +/- 20°.

5.1 Preliminary Calculations

The loads calculated in the computational model were used to size the skin thickness, structural supports, hardware, and mechanical components. The loads used for the structural analysis were the maximum computed loads which occurred at the maximum angle of attack in the computational model, $\alpha = 20°$, and the maximum control surface deflection, $\delta = 20°$. The computational model was developed to accommodate control surface deflection about a hinge line, i.e. the panels of the control surfaces can be oriented to a specified angle. An example of this rotation is shown in Figure 66.

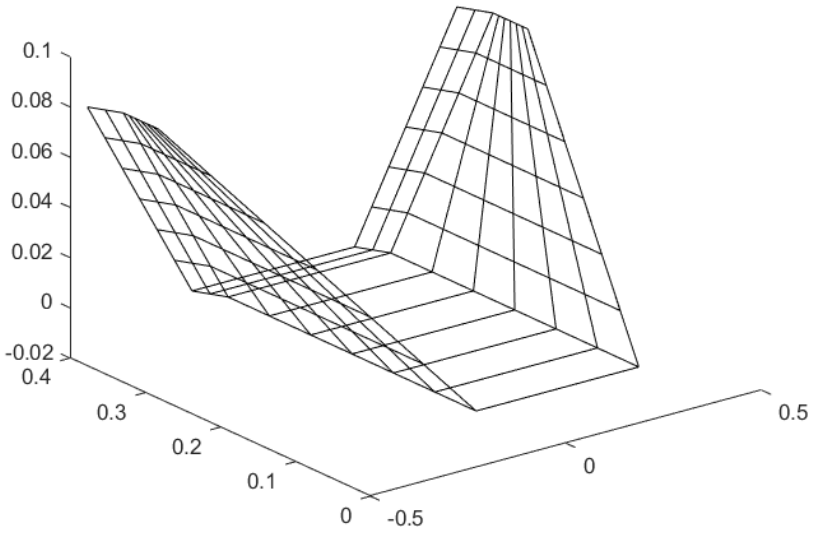


Figure 66. Panel deflection along hinge line

The structural analysis began by determining the resulting shear and bending moments from the applied aerodynamic forces. At this time, it was assumed that the wings would be attached to the body with a fiberglass and epoxy seam. Thus, the wing and body structures were initially analyzed separately. Additionally, because each part consisted of two halves, a seam would be present along the midline of each part. The material yield stress, σ_Y, of the fiberglass and epoxy together was estimated to be 200 MPa.

5.1.1 Body and Wings

It was initially thought that the fiberglass skin would need to be reinforced with a lightweight aluminum frame. The frame was sized using the spanwise loads calculated in the computational model. The model geometry can be seen in Figure 67 and the distribution of these loads is shown in Figure 68.

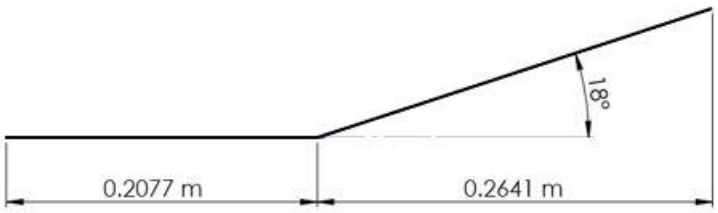
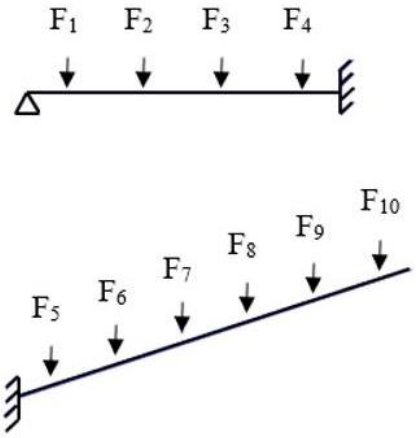


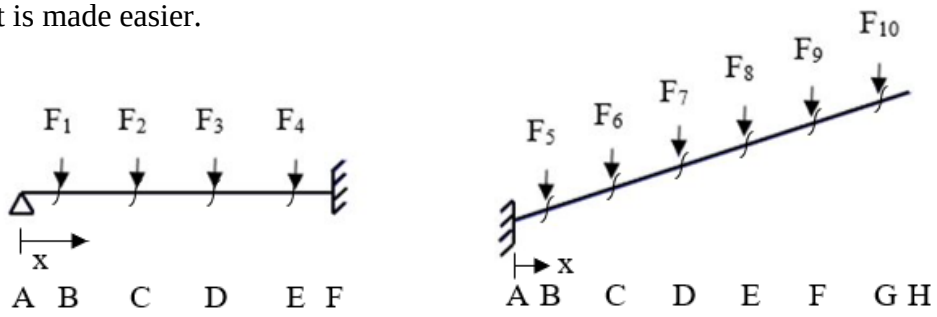
Figure 67. Spanwise model geometry from longitudinal midline



F_1 = 2.0086 N
F_2 = 2.6646 N
F_3 = 2.5293 N
F_4 = 2.0115 N
F_5 = 1.8710 N
F_6 = 1.7670 N
F_7 = 1.6588 N
F_8 = 1.5352 N
F_9 = 1.3842 N
F_{10} = 1.1740 N

Figure 68. Spanwise load distribution

The shear and moment are determined by applying the equations of equilibrium to each segment. If the segment is "cut" as shown in Figure 69, calculation of the resulting shear and bending moment is made easier.



(a) Body (b) Wing

Figure 69. Simplification of beam

The equations of equilibrium can be expressed for regions of concentrated force and moment as shown in Equations 52 and 53 [18]. See Figure 70 for the free-body diagram of the segments.

$$+\uparrow \sum F_y = 0; \qquad V + F - (V + \Delta V) = 0 \tag{52}$$

$$\circlearrowleft + \sum M_0 = 0; \qquad (M + \Delta M) - F\frac{\Delta x}{2} - (V + \Delta V)\Delta x - M = 0 \tag{53}$$

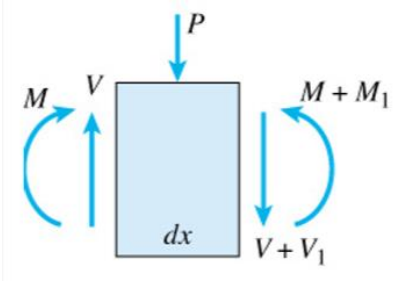


Figure 70. Free-body diagram of beam segment under concentrated load [19].

Thus, for the spanwise segments on the body, the equilibrium equations resolve to:

$$\sum F_y = F_{y,A} + F_{y,F} = 9.214\ N$$

$$\sum M_A = F_1 x_1 + F_2 x_2 + F_3 x_3 + F_4 x_4 = F_{y,F} L \Longrightarrow F_{y,F} = 4.5857\ N$$

$$F_{y,A} = 4.6282\ N$$

$0 < x_1 < 0.007\ m;$

$$\sum F_y = V_B - F_{y,A} = 0 \Longrightarrow V_B = F_{y,A} = 4.6282\ N$$

$$\sum M_0 = -F_{y,A} x_1 + M_B = 0 \Longrightarrow M_B = F_{y,A} x_1 = 0.0324\ Nm$$

$0.007 < x_2 < 0.0692\ m;$

$$\sum F_y = F_{y,A} - F_1 - V_C = 0 \Rightarrow V_C = F_{y,A} - F_1 = 2.6196\ N$$

$$\sum M_0 = 0 \Longrightarrow M_C = F_{y,A} x_2 - F_1 (x_2 - x_1) = 0.1953\ Nm$$

$0.0692 < x_3 < 0.1385\ m;$

$$\sum F_y = F_{y,A} - F_1 - F_2 - V_D = 0 \Longrightarrow V_D = F_{y,A} - F_1 - F_2 = -0.450\ N$$

$$\sum M_0 = 0 \Longrightarrow M_D = F_{y,A} x_3 - F_1 x_3 - F_2 x_3 + F_1 x_1 + F_2 x_2 = 0.1922\ N$$

$0.1385 < x_4 < 0.2007\ m;$

$$\sum F_y = 0 \Longrightarrow V_E = F_{y,A} - F_1 - F_2 - F_3 = -2.5743\ N$$

$$\sum M_0 = 0 \Longrightarrow M_D =$$

$$F_{y,A} x_4 - F_1 x_4 - F_2 x_4 - F_3 x_4 + F_1 x_1 + F_2 x_2 + F_3 x_3 = 0.0309\ Nm$$

$0.2007 < x_5 < 0.2077\ m;$

$$\sum F_y = 0 \Longrightarrow V_F = F_{y,A} - F_1 - F_2 - F_3 - F_4 = -4.5857\ N$$

$$\sum M_0 = 0 \Rightarrow M_F = F_{y,A} x_4 - F_1 x_5 - F_2 x_5 - F_3 x_5 - F_4 x_5 + F_1 x_1 +$$

$$F_2 x_2 + F_3 x_3 + F_4 x_4 = -4.04 \times 10^{-6}\ Nm$$

Here, the reaction force is represented by R and the length by L. The maximum moment occurs when the shear force, V, is equal to zero. From the above calculations, this occurs somewhere between x_2 and x_3 (0.0692 – 0.1385 m). The maximum moment can be estimated at 0.1935 Nm.

The shear and bending moments for the wing section are calculated as would be done for a cantilever beam and shown below.

$+\uparrow \sum F_y = 0; \qquad \sum F_y = F_{y,A} = 9.3902\cos(18) = 8.9306\ N$

$\circlearrowleft + \sum M_0 = 0; \qquad \sum M_A = (F_5x_1 + F_6x_2 + F_7x_3 + F_8x_4 + F_9x_5 + F_{10}x_6)\cos(18) = 1.06\ Nm$

The maximum moments are used to size the structural member that would potentially make up the aluminum frame. Aluminum 6061-T6 was initially selected and its yield stress, σ_y, is 241 MPa. A factor of safety of 3 was applied in Equation 54 to determine the allowable stress. The maximum moment, flexure formula, and section modulus, Equations 55 and 56, are used to determine the cross-sectional shape of the beam.

$$F.S. = \frac{\sigma_{failure}}{\sigma_{allow}} \tag{54}$$

$$\sigma_{max} = \frac{Mc}{I} \tag{55}$$

$$S = \frac{M_{max}}{\sigma_{allow}} \tag{56}$$

Using the maximum moment of 1.06 Nm, which was calculated for the wing root, the beam supporting the load must have a section modulus of 13.20 mm^3. The section modulus is used to determine the cross-sectional geometry of a beam using the relationship in Equation 57. the shear formula in Equation 58 is used to verify the chosen cross-sectional area depending on whether the allowable shear stress is exceeded or not. A beam with a hollow ¼-in. by ¼-in. rectangular geometry and a thickness of 1/8-in. was chosen for the aluminum frame because it had a section modulus of 56.8 mm^3, which is sufficiently greater than what was required. The area moment of a hollow rectangle formula needed for the calculations is shown in Equation 59.

$$S_{required} = \frac{I}{c} \tag{57}$$

$$\tau_{allow} \geq \frac{V_{max} Q}{It} \tag{58}$$

$$I = \frac{bd^3 - hk^3}{12} \tag{59}$$

The skin is assumed to carry the shear stress and some axial stress while stiffeners carry the axial stress due to the bending moments and shear forces on the vehicle during flight. The maximum bending moment along the length of the wings and lifting body fuselage can be used to determine the size and number of stiffeners needed to reinforce the skin and the required skin thickness can be calculated with respect to the stiffener design. Preliminary results for the shear and bending moment were determined by first simplifying the loads calculated in the computational model to a simple beam. Table 3 lists the loads acting on the lifting body. The total force on the lifting body fuselage was simplified to a distributed load and the moment calculated using Equation 60. The maximum moment along the length of the lifting body fuselage is calculated to be 1.775 Nm at half of the vehicle reference length.

Component	**Vertical Load**
Lifting Body Fuselage	16.4194 N
Wing	9.60 N
Elevon	2.469 N
Body Flap	4.1662 N

Table 3. Loads acting on vehicle

$$M_{max}\left(x = \frac{L}{2}\right) = \frac{w}{2}(Lx - x^2) \tag{60}$$

The cross-sectional geometry was simplified to a circle with a width of 0.31 meters and a height of 0.26 meters to solve for a preliminary value of the skin thickness. The stiffeners are simplified to concentrations of area that are called booms. Between the booms, the normal stress

varies along the segment of skin. This segment of skin can be represented by booms on either side such that the same force and moment acting on the skin is acting on the booms. This method of analysis allows the assumption that the booms carry the normal stresses and the skin carries the shear stress. The normal stress is calculated using Equations 61 and 62. The boom area is represented by Equation 63.

$$I_{zz} = I_{yy} = \sum_{i=1}^{n} z_i^2 B_i \tag{61}$$

$$\sigma_x = \left(\frac{M_y I_{zz} - M_z I_{yz}}{I_{zz} I_{yy} - I_{yz}^2}\right) z + \left(\frac{M_z I_{yy} - M_y I_{yz}}{I_{zz} I_{yy} - I_{yz}^2}\right) y \tag{62}$$

$$B_i = \frac{t_D b}{6}\left(2 + \frac{\sigma_2}{\sigma_1}\right) + \frac{t_D b}{6}\left(2 + \frac{\sigma_n}{\sigma_1}\right) \tag{63}$$

Equations 61 – 63 are used to solved for the number of stiffeners needed at the location of the maximum moment. The location of each stiffener was chosen to be along the seam lines, i.e., the places where the rigid molds would have flanges to fasten together. The stiffener area at these locations was chosen to be a rectangle of arbitrary height and width of 0.0015 meters and 0.254 meters, respectively, to represent layers of fiberglass and resin making up the flange. At these locations, the seam must be able to amass the total bending moment because the skin is ineffective. The net area moment of inertia, I_{yy}, was computed considering the stiffeners to be lumped masses.

A preliminary number of four stiffeners was chosen to begin the analysis. Assuming the fiberglass and resin composite had a yield stress of 200 MPa, the allowable stress for the design was 67 MPa for a safety factor of 3. Equation 62 was used to determine the direct stress in each stiffener produced by the bending moments M_y and M_z using the inertia calculated in Equation 61.

$n = 4$: $\sigma_{allow} = \frac{\sigma_{yield}}{3} = 67\ MPa$

$$B = \frac{t_D b}{6}\left(2 + \frac{\sigma_2}{\sigma_1}\right) \times 2 = \frac{(0.0015\ m)(0.224\ m)}{6}(2 + 0) \times 2 = 9.86 \times 10^{-4}\ m^2$$

$$I_{yy} = Bz^2 = 2(9.86 \times 10^{-4}\ m^2)(0.13)^2 = 3.333 \times 10^{-5}\ m^4$$

$$\sigma_x = \frac{M_y}{I_{yy}} z = \frac{1.775\ Nm}{3.333 \times 10^{-5}\ m^4}(0.13\ m) = 6923\ \frac{N}{m^2} = 0.0069\ MPa$$

The direct stress produced is much smaller than the yield stress of the material. Because the result was so much smaller than the yield stress, the minimum skin thickness, t_D, was determined under maximum load.

$$\sigma_{yield} = 200\ MPa$$

$$I_{yy} = \frac{M_y}{\sigma_{xx}} z = \frac{1.775\ Nm}{200\ MPa}(0.13\ m) = 1.1538 \times 10^{-9}\ m^4$$

$$B_1 = \frac{I_{yy}}{z^2} = \frac{1.1538 \times 10^{-9}\ m^4}{(0.13\ m)^2} = 6.8269 \times 10^{-8}\ m^2$$

$n = 4$: $B_1 = \frac{t_D b}{6}(2 + \nearrow^0) \times 2 \Longrightarrow t_D = \frac{6B_1}{2b} = \frac{6(6.8269 \times 10^{-8}\ m^2)}{2(0.224\ m)} = 9.143 \times 10^{-7}\ m$

$n = 2$: $t_D = 4.572 \times 10^{-7}\ m$

The minimum skin thickness is very small, so it was assumed the glider could be structurally sound as designed.

5.1.2 Control Surface Hinges

The hinge rods for the elevons and body flap were sized with respect to the torque corresponding to the load at maximum deflection. The torque was calculated by multiplying the force value by the distance from the hinge to the point where the linkage is fixed to the control surface, this was the effective distance. Equation 64 was used to determine the applied torque.

$$M = Fx \tag{64}$$

Using the forces calculated in the computational model, in Table 3, the torque on the elevon was calculated to be 0.119 Nm for a distance, x, of 0.0120 meters. The torque on the body flap was calculated to be 0.3240 Nm for a distance of 0.0195 meters.

The moment about the elevon aerodynamic center was calculated using Equation 51 and was found to be 0.0247 Nm. Equation 43 was used to determine the section modulus for the aluminum hinge rod, where $S_{req'd} = 3.07e^{-2}$ mm^3. Both hollow and solid rods were considered, and Equations 65 and 66 were used to calculate the area moment of inertia for the solid and hollow rods, respectively.

$$I_{xx} = I_{yy} = \frac{\pi d^4}{64} \tag{65}$$

$$I_{xx} = I_{yy} = \frac{\pi}{64}(d_{outer}^4 - d_{inner}^4) \tag{66}$$

A hollow rod with an outer diameter, d, of ¼-inch was found to have a section modulus of 18.03 mm^3, which is much higher than necessary. Thus, it was assumed a solid rod with the same dimensions would also be sufficient.

The allowable shear stress, τ_{allow}, can be calculated from Equation 67 using the torsion formula in Equation 68. The allowable shear stress was found to be 40 MPa.

$$\tau_{allow} = \sqrt{\left(\frac{\sigma}{2}\right)^2 + \tau^2} = \sqrt{\left(\frac{Mc}{2I}\right)^2 + \left(\frac{Tc}{J}\right)^2} \tag{67}$$

$$\tau = \frac{Tc}{J} \tag{68}$$

Here, T is the resultant internal torque acting on the cross section and c is the outer radius of the shaft. The polar moment of inertia, J, can be calculated using Equation 69 for a solid shaft and Equation 70 for a tubular shaft [18].

$$J = \frac{\pi}{2} c^4 \tag{69}$$

$$J = \frac{\pi}{2} \left(c_{outer}^4 - c_{inner}^2 \right) \tag{70}$$

For a solid shaft, the torsion formula yielded a result of 0.03141 MPa which is much smaller than the allowable shear stress.

5.2 Solidworks Static Structural Analysis

The Solidworks model was initially created to model the image shown in Figure 71. The Solidworks model was created by importing the three dimensional views onto planes in Solidworks. Each plane was bisected at the same location along the x-axis. Points were sketched on the outline of the image in the three planes and lofted together to create a 3D solid body. The initial geometry was updated during the aerodynamic analysis to have a wider lifting body fuselage and a wing incidence of -2°.

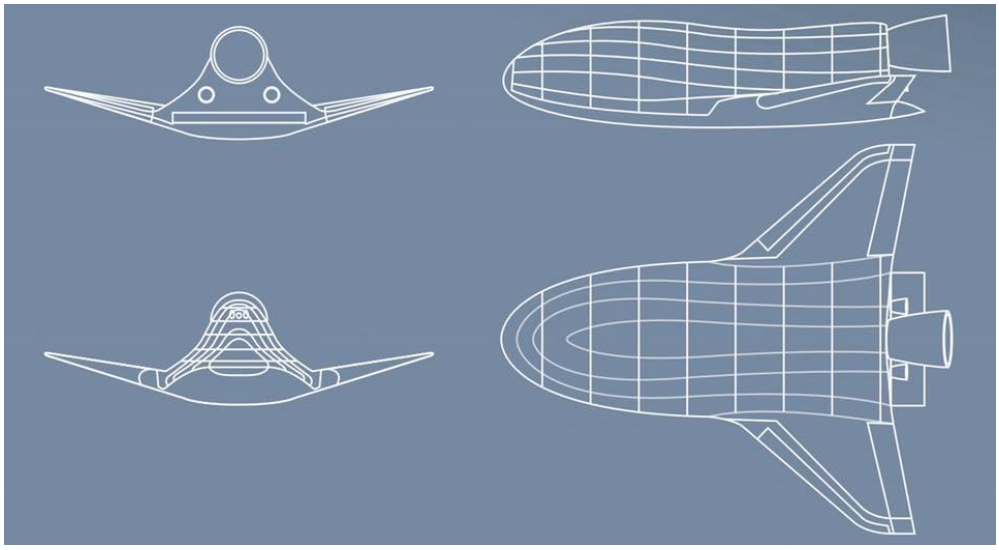


Figure 71. Initial lifting body glider geometry

Final verification of the structural model was conducted using Solidworks static simulations. The computed loads in MATLAB were applied to their respective surfaces in the Solidworks model and analyzed. Before the simulations could be conducted, the model was updated to have the anticipated structure and geometry. Primarily, the Solidworks model was

hollowed out to have a skin thickness of 1.5 mm. Then, the control surfaces were cut from the solid body and shaped to be aerodynamically suitable. The elevons were separated from the wings by 1 mm and their leading edges rounded to accommodate for the control surface deflection. Similarly, the body flap was separated from the solid body and placed a distance from the body suitable for deflection without interference in the ± 20° range. The resulting Solidworks model is shown in Figure 72.

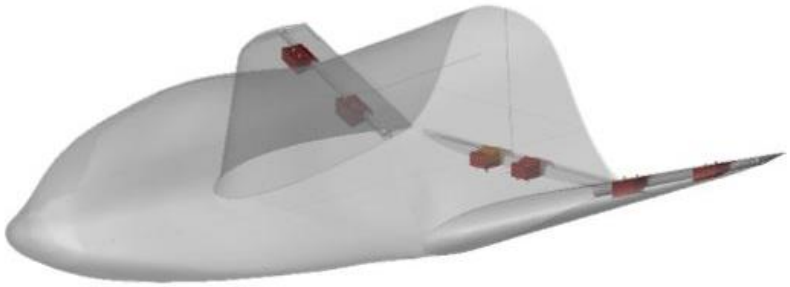


Figure 72. Final Solidworks model

The wing was analyzed separately from the body due to meshing issues from the complex geometry. The wing was fixed at the root, where it would attach to the body, and where there were any fasteners and loaded with the anticipated aerodynamic loads which were calculated in the computational model. The loads were applied to the lower surfaces of the wing and elevon. The elevon was deflected to verify the stress reaction on the hinge. Results from the first simulation indicated the aluminum frame might be unnecessary because the maximum stress was very low. The results from the initial simulation with the aluminum frame are shown in Figures 73 – 75. A second simulation was executed without the aluminum frame to determine if the fiberglass and epoxy composite was strong enough. The results of that simulation are shown in Figures 76 – 78. It is shown that the maximum stress of 3.251 MPa is much lower than the yield stress value of 200 MPa and the safety factor for the configuration without the aluminum frame was 61. Therefore, the design without the aluminum frame was accepted.

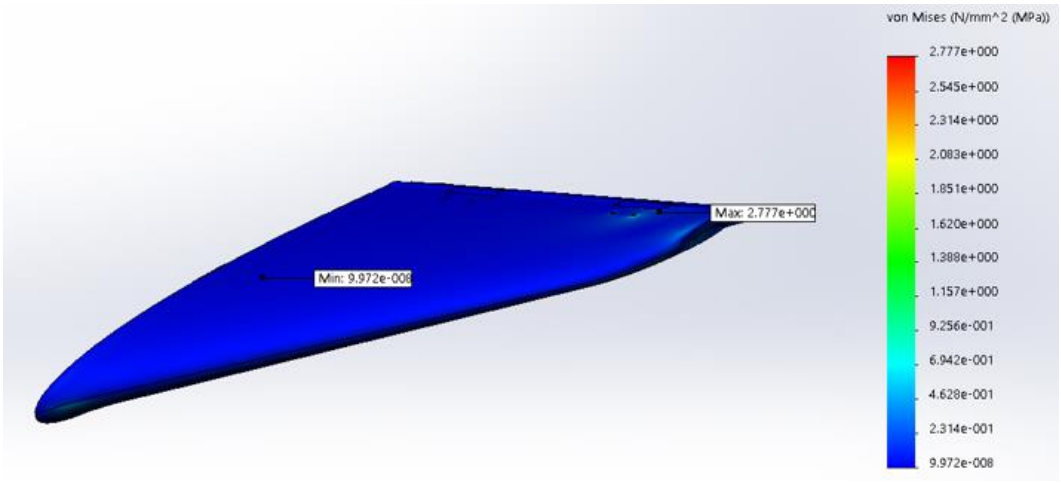


Figure 73. Wing with bar stress results

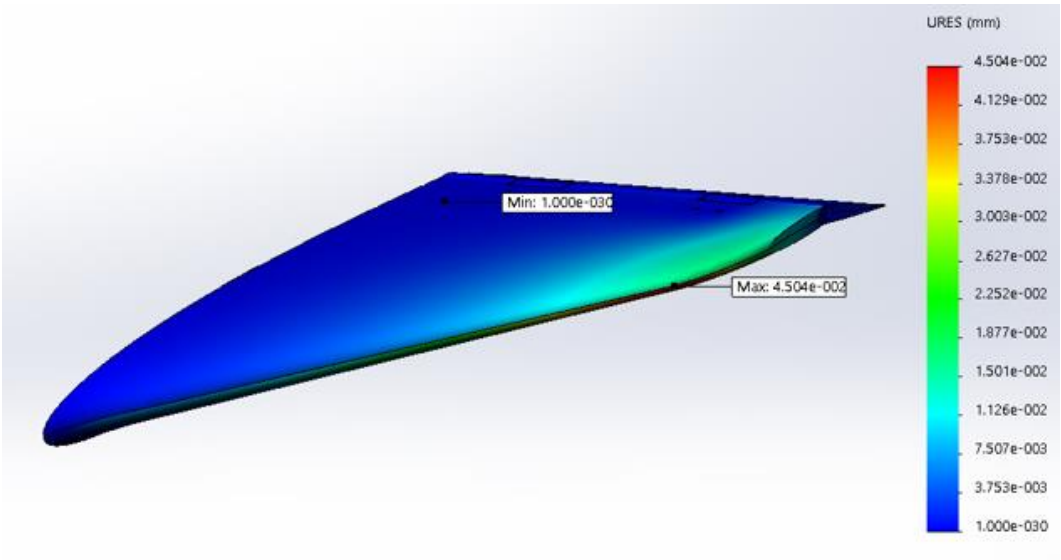


Figure 74. Wing with bar displacement results

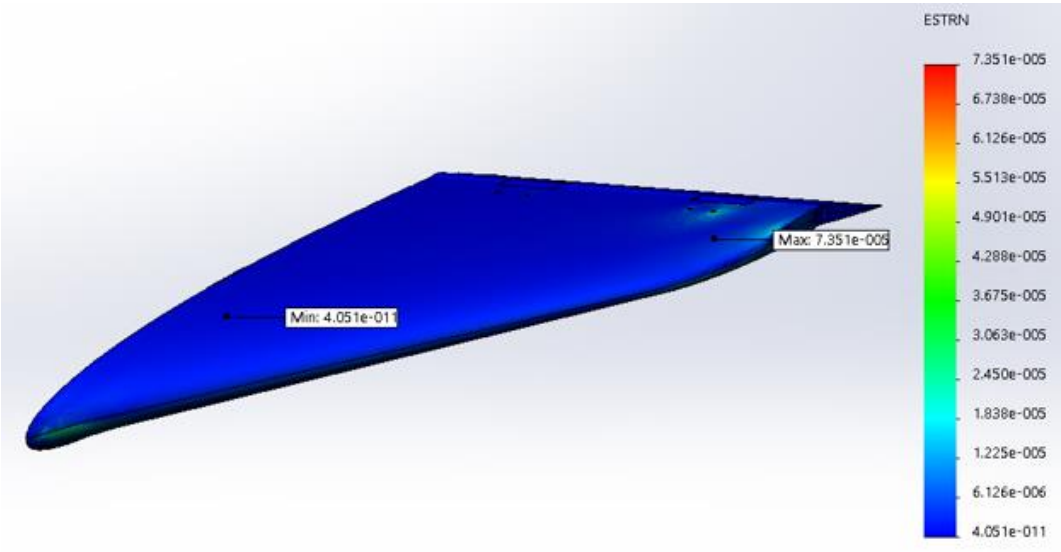


Figure 75. Wing with bar strain results

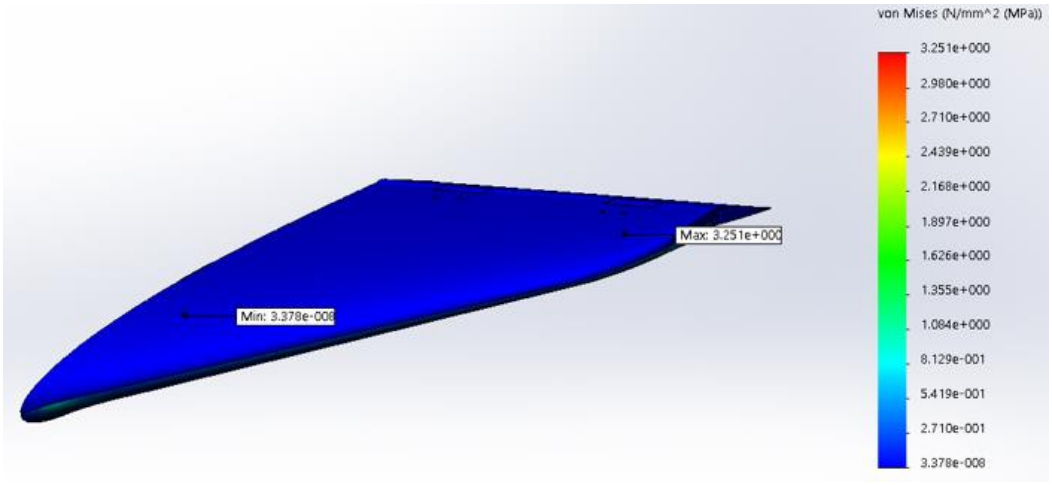


Figure 76. Wing without bar stress results

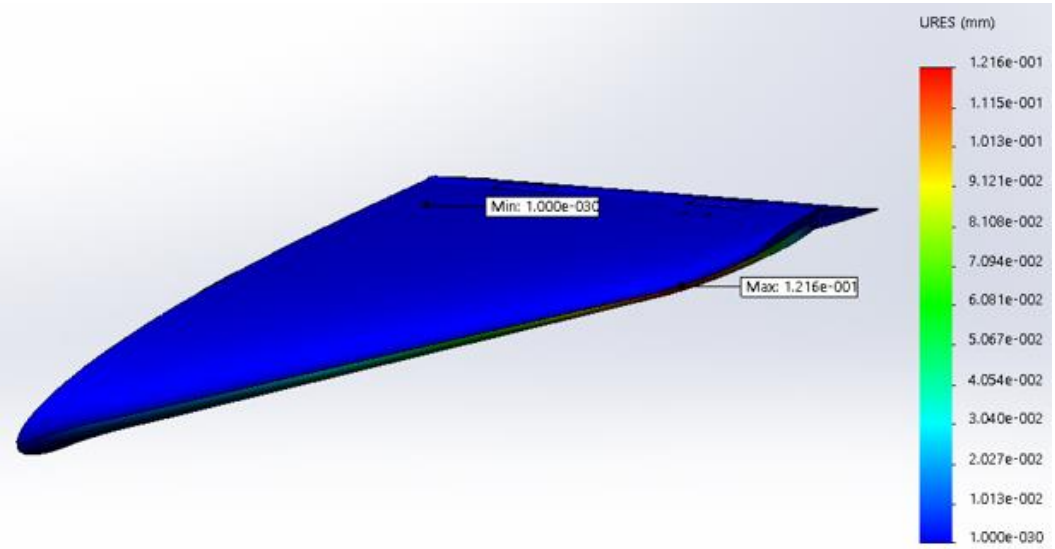


Figure 77. Wing without bar displacement results

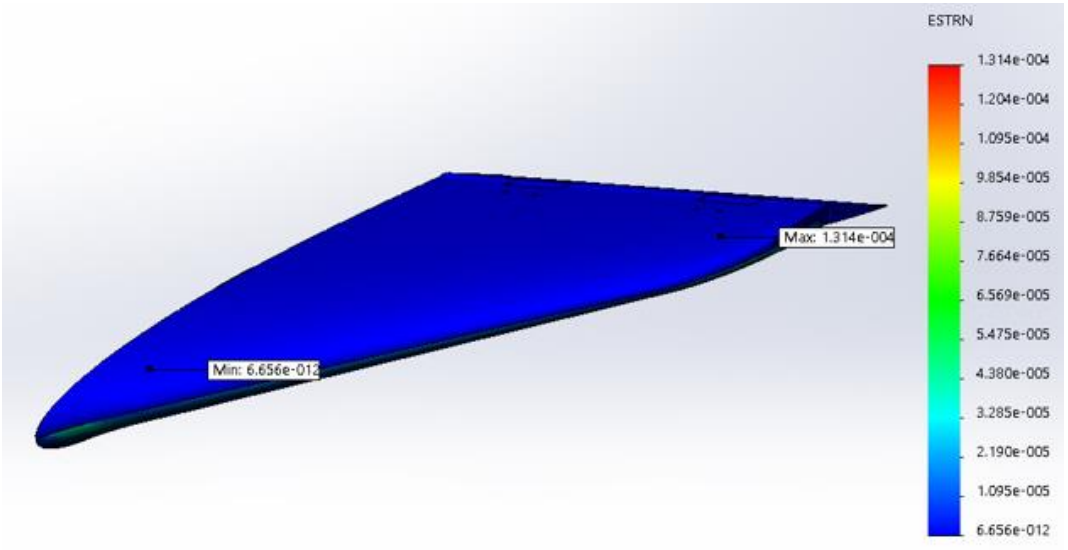


Figure 78. Wing without bar strain results

The lifting body fuselage was analyzed separately from the wing. It was fixed at the wing root and where there were fasteners. Aerodynamic loads calculated using the computational model were applied to the lower surfaces of the body and body flap. The body flap was deflected to ensure the simulation was accurately calculating the stress distribution. The maximum stress was found at the wing root leading edge and was a value of 0.3018 MPa which is much smaller than the yield stress of the composite at 200 MPa. The static simulation results for the body are shown in Figures 79 – 81.

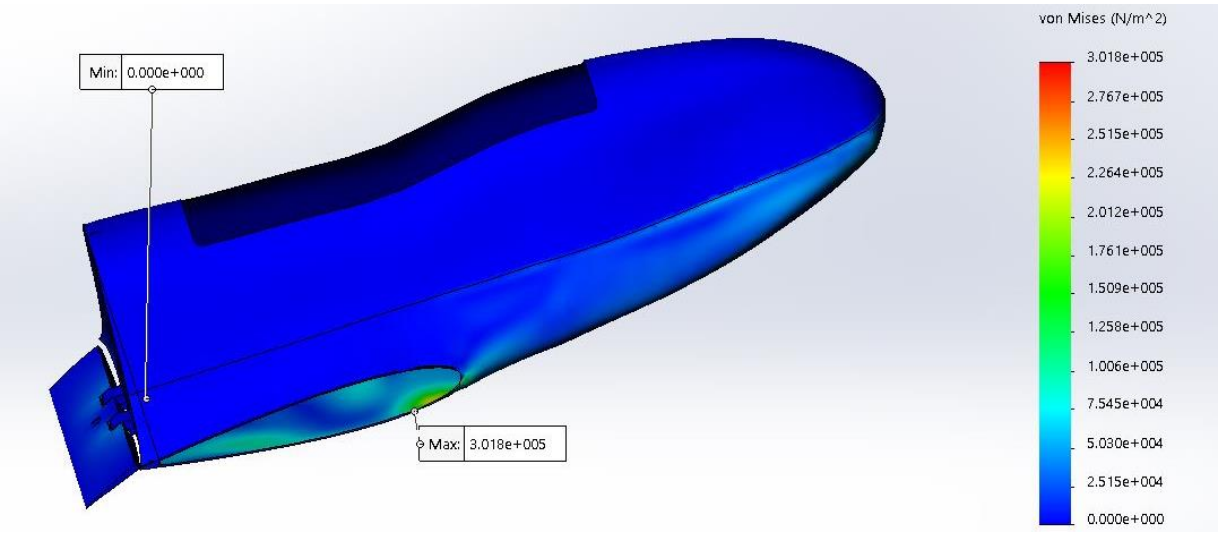


Figure 79. Glider body stress results

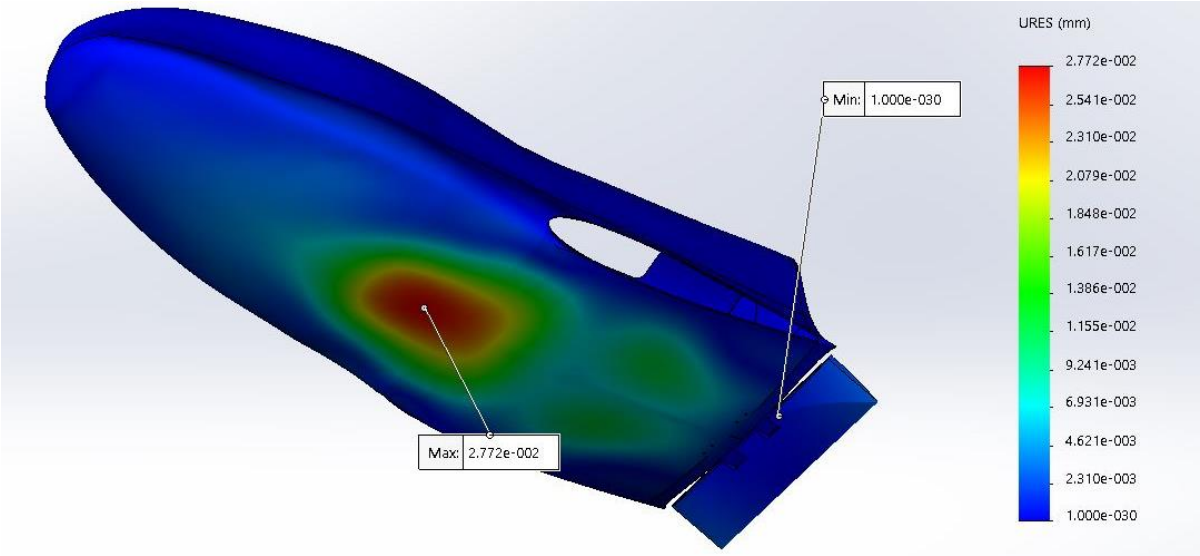


Figure 80. Glider body displacement results

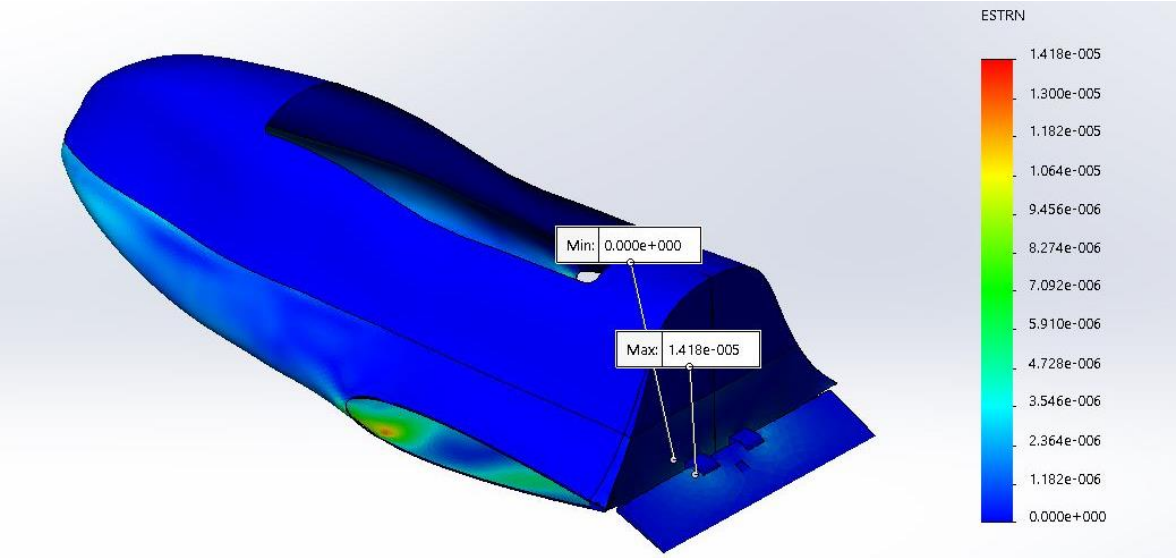


Figure 81. Glider body strain results

The hinge blocks were made from 3D printed PLA and were modified throughout the simulation process to be more robust. Preliminary designs had the wing hinge blocks at a greater distance from each other. Simulations of this configuration identified an issue with the material between the hinge blocks having an unwanted reaction at the midpoint. Due to the unwanted stress, the hinge blocks on the wing were moved closer together and the issue was mitigated. The hinge blocks were solid bodies in Solidworks but are not so on the physical model. 3D printed items typically have an outer wall of an arbitrary thickness and an inner mesh that comprises the support structure. A compromise between stiffness and weight was assessed and the hinge blocks were ultimately designed to have a 2 mm thick shell with a 50% inner mesh. The hinge blocks were not analyzed alone but were included in the simulations for the wing and body. During these simulations, a large amount of stress was found on the body flap hinge shaft clamping collars. To mitigate this, the body flap hinge blocks were designed to house the shaft collars for added support. The hinge block designs are shown in Figure 82.

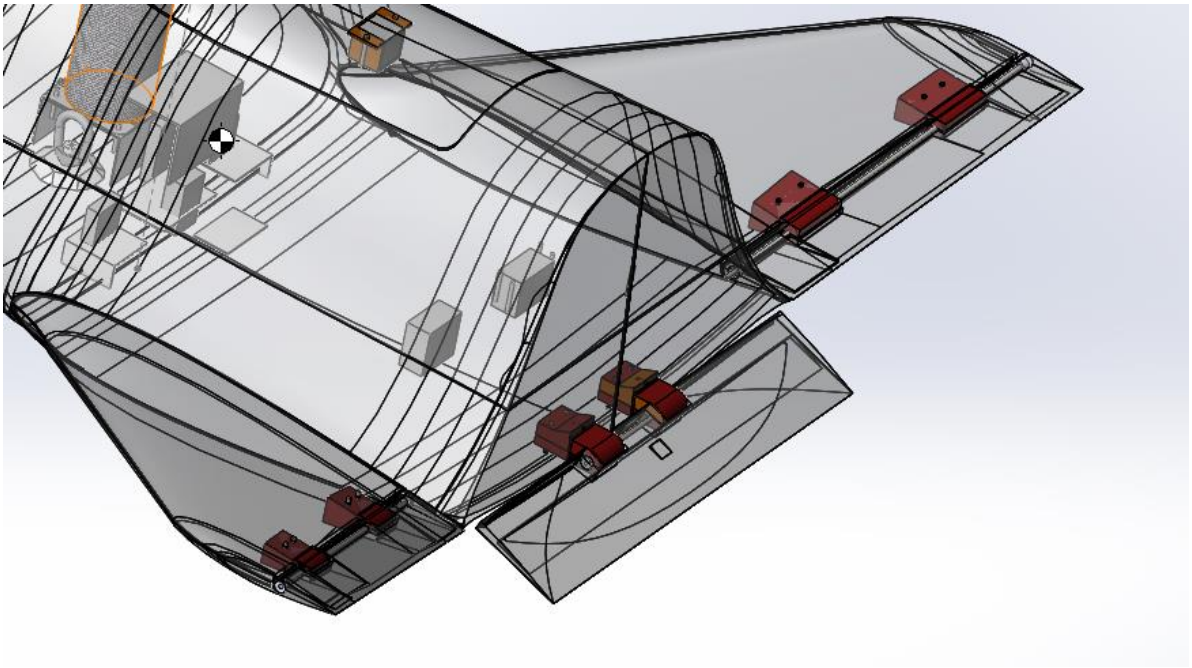


Figure 82. Hinge blocks in Solidworks Model

A critical component of the structure is the parachute tie down. The anticipated force on this component was calculated using Equation 71 where A_s is the parachute surface area and d is the diameter.

$$F_{D,parachute} = \frac{1}{2}\rho v^2 C_D A_s \tag{71}$$

The initial force when the parachute opens and the glider is moving at maximum velocity, 18m/s, was calculated to be 541.7 N. It is possible to solve for the rate at which the glider descends once the parachute is deployed, shown in Equation 72. For a parachute area of 1.82 m^2, the rate of descent was calculated to be 2.153 m/s. This means the parachute must be deployed at an altitude greater than 10 meters for the glider to decelerate to a reasonable speed before landing.

$$v = \sqrt{\frac{mg}{\pi\rho C_D d^2}} \tag{72}$$

A Solidworks structural analysis was performed on a representation of the tie down component attached to the composite material of the lifting body glider. It was necessary to

create a representation of the composite material due to the complexities of the curvature of the lifting body fuselage making meshing difficult. The representation for the static simulation is shown in Figure 83.

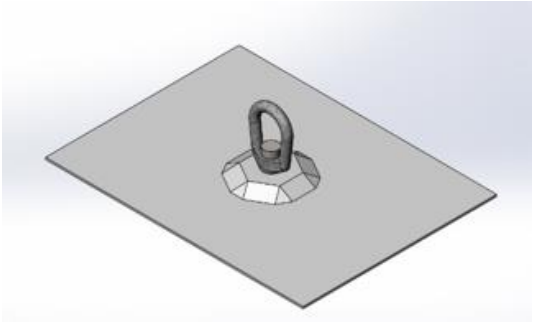


Figure 83. Parachute tie down Solidworks simulation model

The rectangular base represents the skin of the glider. The geometrical housing is made of fiberglass and resin to help distribute the stress from the tie down. The force is applied to the galvanized steel eye nut, which is bolted through to a nut encased in the geometrical housing composite. A force of 541.7 N was applied to the top of the eye nut for the simulated maximum load. The only fixture to be applied was around the edges of the composite skin. The simulation results are shown in Figures 84 – 86. A safety factor of 2.9 was found for this configuration, therefore, the tie down design was determined to be acceptable.

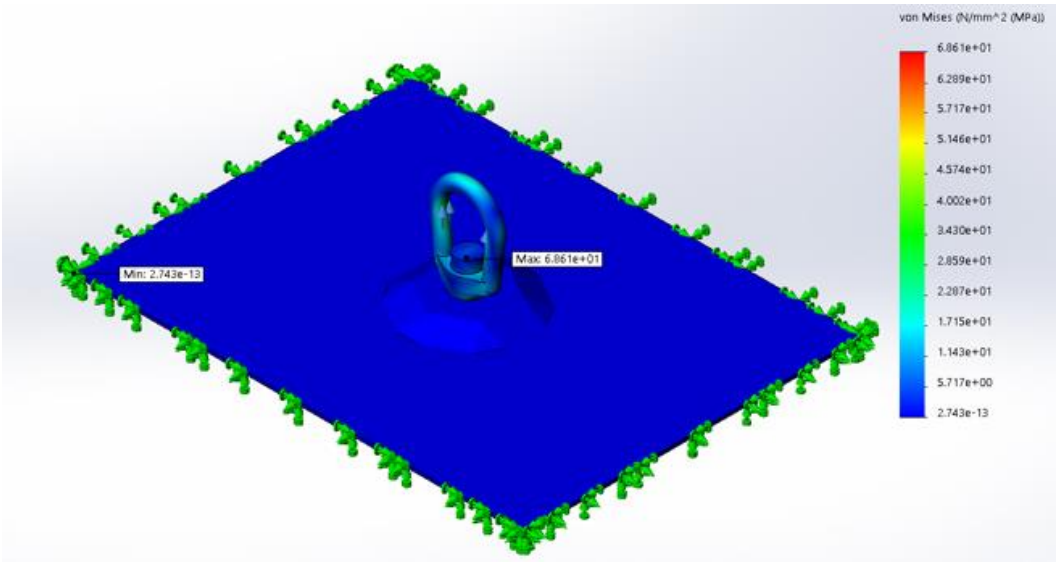


Figure 84. Tie down design stress results

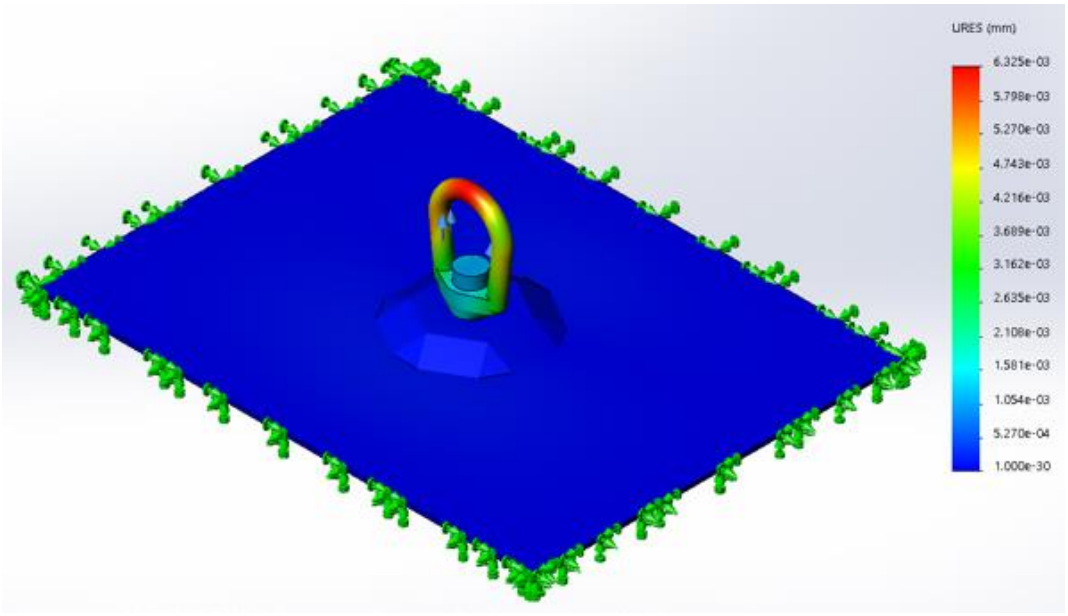


Figure 85. Tie down design displacement results

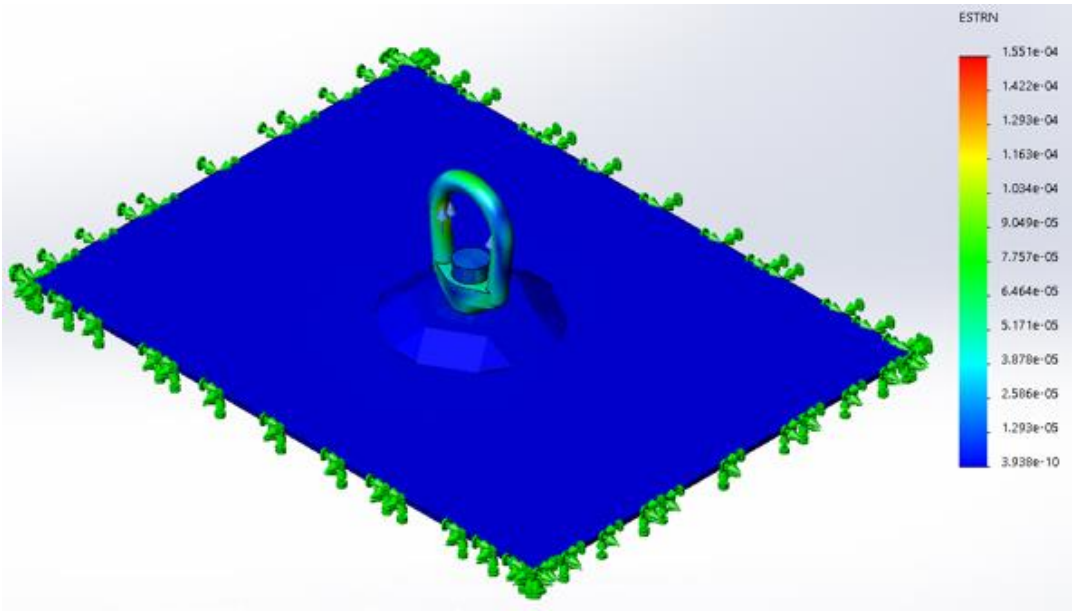


Figure 86. Tie down design strain results

A 3D printed housing was designed for the servo release mechanism. The release mechanism consists of a servo connected to a pin which holds a linkage to the high altitude balloon chord. The housing is made of two pieces, a lower piece which is load bearing and an upper piece that constrains the components to maintain their positions. The lower piece is shown in Figure 87.

Figure 87. Load bearing piece of release mechanism housing

The housing was tested under the maximum weight of the glider, an estimated 6 kg multiplied by the maximum value of gravity at 9.81. The flat face was fixed where it was to be bolted to the glider and fixed where the servo was bolted through the housing. Results from the simulation indicated the greatest stresses were found around the extrusion cut for the servo. Of this area, the maximum stress was calculated to be 5.05 MPa while the yield stress of the housing was estimated to be 63.5 MPa. This resulted in an acceptable safety factor of 12. An image of the stress distribution in the housing is shown in Figure 88.

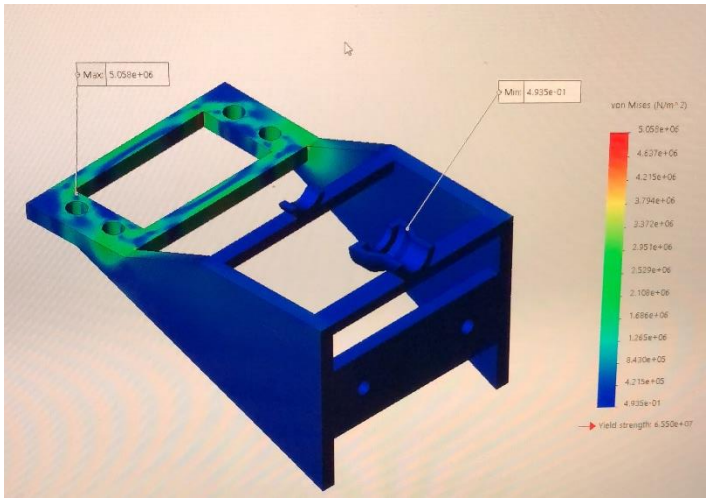


Figure 88. Servo release mechanism housing stress results

CHAPTER 6. FABRICATION

6.1 Foam Operations

The glider is made of a fiberglass and resin composite which were shaped into the glider geometries using high density foam molds. Upon the initial longitudinal stability results, 3D models were prepared in Solidworks and submitted for fabrication at the Kennedy Space Center Foam Operations facility. The foam operations can be seen in Figures 89 and 90.

Figure 89. KSC Foam Operations machining

Figure 90. KSC Foam Operations (elevons and body flap)

The foam models were sections of the glider that were to be cut from high density foam and used to make rigid molds from fiberglass and epoxy resin. The glider was segmented by the body, wings, and control surfaces, with each segment having a top and bottom section. The foam

molds, shown in Solidworks in Figures 91 and 92, were prepared for the rigid molding by sanding and painting the outer surfaces until smooth.

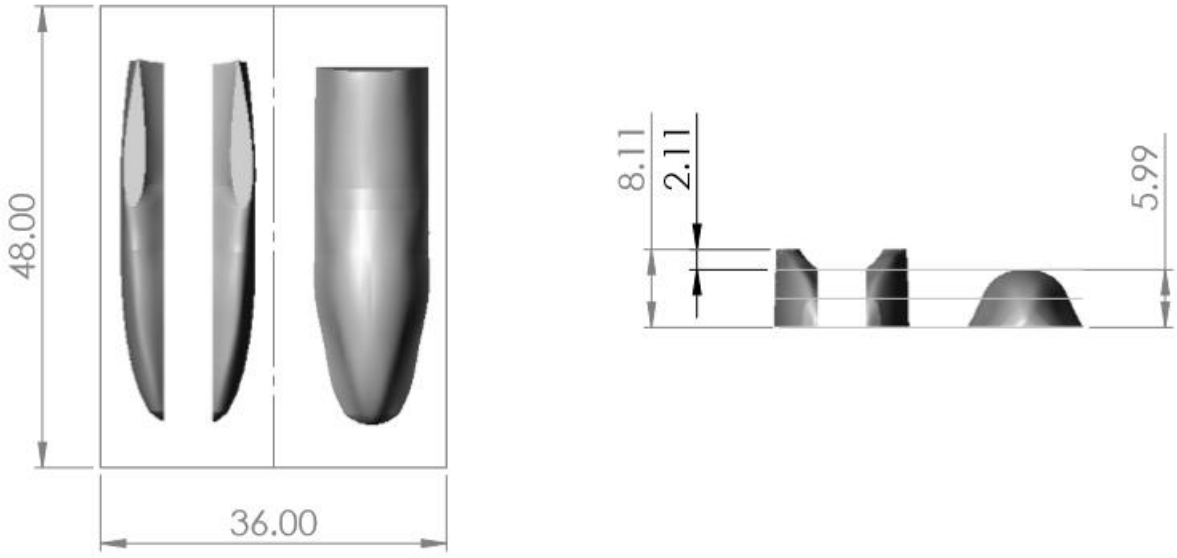


Figure 91. Solidworks drawing of foam mold body sections (inches)

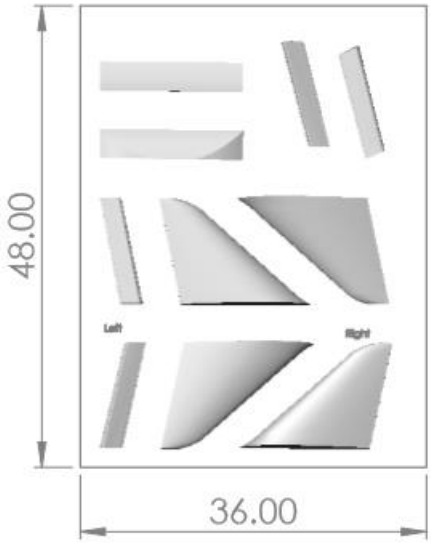


Figure 92. Solidworks drawing of foam mold wing and control surface sections (inches)

The images above show the segments submitted to the foam operations for carving. Each segment was cut along its half line to create a flat "bottom" surface, i.e., something to lay flat while the geometry is carved out of the foam blocks. Each foam mold remained separate except the bottom of the glider body. This mold in particular had to be cut along its longitudinal midline to properly carve out the wing root geometry, working around the curvature of the body. This

portion was then glued together at KSC Foam Operations to become one continuous mold of the bottom half of the body, shown in Figure 93.

Figure 93. Bottom half of glider body modeled in high density foam

6.2 Materials

E-glass and polyester molding resin were used to create the glider and rigid molds. The foam was mounted to flat surfaces, sanded smooth, and painted a paint gun to create a smooth surface prior to making the rigid molds. The painting and sanding procedure was repeated until the surfaces had minimal texture. Once the foam parts were sufficiently smooth, a glossy coat of paint was applied as an outer coat. The surface was then prepared for molding using a demolding agent which would allow for the removal of the composite layers from the mold. A photo of the foam molds with the final coat of paint are shown in Figure 94.

Figure 94. Wing and control surface foam molds painted and prepared for rigid molding

Fasteners used in the fabrication of the glider are predominantly steel with the exception of the set screw collars used to secure the hinge rods, which are made of aluminum. The housings for the control surface hardware are made of fiberglass and were designed into the control surfaces. The control linkage to the body flap is made of steel while the linkages to the elevons are nylon. Due to the unique geometry of the glider, rigid linkages to the ailerons would prove difficult to install. Thus, flexible nylon linkages were used. A complete bill of materials for the glider fabrication is shown in Table 4.

Component	**Use**
Fibre Glast Style 120 E-Glass	Part/mold fabrication
Fibre Glast Polyester Molding Resin	Part/mold fabrication
Fibre Glast MEKP (Methyl Ethyl Ketone Peroxide)	Curing agent
1/8" Stainless Steel Precision Shafting	Hinge rod
Aluminum Set Screw Collars	Hinge rod fasteners
Galvanized Steel Eye Nut - for Lifting	Parachute tie down
High-Strength Steel Hex Nut	Parachute tie down fastener
18-8 Stainless Steel Hex Nut 6-40 Thread Size	Fasteners for internal components
18-8 Stainless Steel Button Head Hex Drive Screw 5-40 Thread Size	Hinge block fasteners
18-8 Stainless Steel Narrow Hex Nut 5-40 Thread Size	Hinge block fasteners
18-8 Stainless Steel Socket Head Screw 5-40 Thread Size	Parachute fasteners
3/4" High Steel Control Horns w/Nylon Bushing	Control surfaces
Linkage - steel	Control surfaces
Linkage - nylon	Control surfaces

Table 4. Glider bill of materials

6.3 Methodology

6.3.1 Molding Process

A series of rigid body molds were made from the foam molds. Demoldant was applied to the painted surface in even layers using a paint brush. Once dry, the surface was ready for the rigid molding process. The polyester resin was measured out and mixed with the appropriate amount of hardener. It was important to quickly apply the mixture because the curing process is very fast. Once the resin was applied, a fiberglass sheet was laid on the foam mold. The fiberglass was pressed into the resin to fully embed it in the liquid. Once the air bubbles had been removed, a second layer of fiberglass was placed on the first such that the weaves were at an angle of approximately 45°. The composite took about a day to harden into the rigid molds. An image of the body flap rigid mold is shown in Figure 95.

Figure 95. Rigid mold for the body flap removed from the foam mold

The process to create the glider geometry was the same. The rigid molds were prepared with the demolding agent. A resin and curing agent mixture was applied to the rigid molds and fiberglass was embedded in the liquid to create the final mold used for the glider geometry. The

fiberglass sheets were again layered at approximately 45° angles. For the final molds, three layers of fiberglass were used such that the skin was about 1.5 mm thick. Figure 96 shows the curing composites in the rigid molds for the upper body, door, and wing.

Figure 96. Final fiberglass mold process

6.3.2 Assembling the Glider

The fiberglass components were joined together along their “cut lines” by fiberglass seams. The seams were made of the same fiberglass and resin mixture and were about 4 inches wide, allowing 2 inches to extend into each component. Initially, three layers of fiberglass and resin were installed but after the first flight tests two additional layers were added. Referring back to the structural analysis that was performed to determine the size and number of stiffeners, the required area of the seams was very small. This meant the seams did not need to be very thick. However, the first test caused some damage to the vehicle upon landing and additional

seams were added for safe measure. Figure 97 shows the two portions of the bottom half of the body after they were joined together. Figure 98 shows the disassembled glider.

Figure 97. Lower half of body sections joined by seam

Figure 98. Disassembled glider

The wings were added to the body using the same seaming method. The SolidWorks model was used to design a wing guide that would position the wing at the proper dihedral and incidence angle to the body. The wing guides were fabricated using a 3D printer and positioned at the appropriate location relative to the wing root for wing installation. The dihedral angle of

the wing midline was verified both before and after wing installation, shown in Figure 99. To support the resultant stress from the wings, a seam was added around the cross section of the lower half of the body at the root of the wing leading edge.

Figure 99. Wing installation using 3D printed wing guide

Before the upper and lower components were joined, extrusions were cut for the control surface hardware. These extrusions were made to insert the hinge blocks, which would house and fix the hinge rods in the necessary locations. Figure 97 shows the extrusions made for the body flap hinge blocks and Figure 100 shows those made for the wing hinge blocks.

Figure 100. Wing extruded cuts for hinge blocks

In some cases, the two halves of each component were difficult to line up for the seaming process. The most difficult pieces to seam were the upper and lower body parts. It is thought that the warping occurred because of how large the parts were and how the unique curvature of the body made the molding process difficult. In some areas, there was a centimeter difference between the upper and lower cut lines. To mitigate the issue, the upper and lower body pieces were seamed in sections using clamps to align the section being seamed. The initial seam consisted of two layers, and once the entire upper and lower pieces were joined a continuous third layer was added. Figure 101 shows the clamping method for the seaming process.

Figure 101. Clamping the upper and lower body pieces during the seaming process

6.3.3 Fiberglass Housings

To cut down on weight, some of the housings were made using the fiberglass and resin composite. Housings made of fiberglass included those for the wing and body flap servos, hinge blocks, and hinge rods. The simplest housings were those for the hinge blocks, which essentially consisted of constructing a boundary to fix the hinge blocks along one or two sides. The hinge

blocks were bolted into the skin through holes drilled to pass the fasteners through, thus the fiberglass housings were designed to prevent small translations of the hinge blocks.

The servos were installed in fiberglass housings that were made to accommodate the size of the servos. The housings were platforms with extrusions cut on the flat face to house the servo body and necessary bolts. These were installed into the glider by created a fiberglass flange on either end and using the seaming method to fix the housing to the glider skin. Figure 102 shows a housing for one of the wing servos.

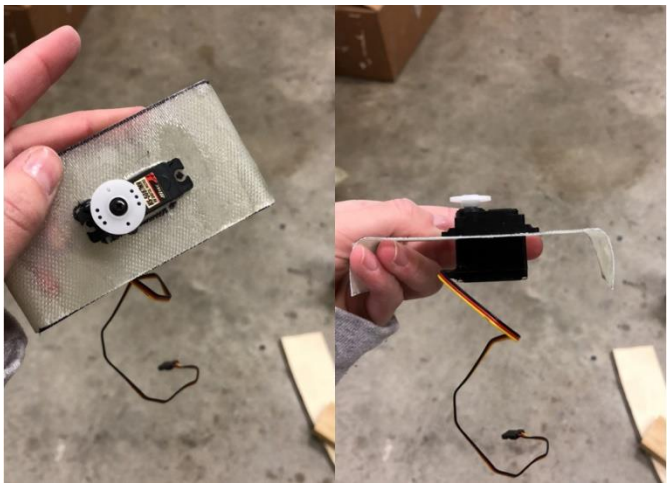


Figure 102. Fiberglass housing for wing servo

The most complex housing was for the hinge rods. SolidWorks simulations were conducted to evaluate the fixture design for each of the control surfaces. Results from these simulations indicated that the safety factor fell below the required value if the hinge rods and shaft collars were unsupported. Design changes through trial and error revealed that a 2 mm thick fiberglass casing would bring the safety factor to the required value of 3. This casing needed to cover the hinge rod and support the shaft collars, while still allowing rotation about the shaft. The most difficult casing to fabricate was for the hinge rods. A layer of plastic sheeting was wrapped around the rods to prevent the rods from touching the fiberglass and resin mixture. The fiberglass and resin mixture was wrapped around the plastic covered rod, and a final layer of

plastic was wrapped around the composite as it cured. The purpose of the outer plastic layer was to compress the composite around the rod as much as possible so it would retain its shape during the curing process. Once the casing had cured, the rod was removed and the casing was cut down to be installed into the control surfaces. However, the rod was inserted into the casing once again while the casing was being installed into the control surfaces to ensure the geometry of the hinge line remained intact. Once the hinge rod casing was installed, it was determined that the shaft collars function improved if they were epoxied into the control surfaces. Figure 103 shows the upper half of an elevon with the hinge rod and casing visible.

Figure 103. Elevon hinge rod and casing

6.3.4 Installing the Control Surfaces

The control surface molds were modified to accommodate the hinge blocks and hinge rods before they could be installed into the glider. SolidWorks simulation results indicated that the hinge block locations were important to the structural integrity of the control surfaces. If placed too far apart, the wing would incur a bending moment at the leading edge midpoint. To ensure the hinge blocks were adequately placed, pdf drawings were made of the SolidWorks model and printed out at a 1:1 ratio and used as a guide for the cuts. Reciprocal cuts were made

in the wing. Once the cuts were made, fiberglass walls were constructed to close the areas around the hinge blocks. The process of building up the fiberglass around the hinge block cuts is shown in Figure 104.

Figure 104. Constructing hinge block housings into the control surfaces

The hinge rod casings can also be seen under construction in Figure 104. The fiberglass walls were cut down so that the upper and lower halves of the control surfaces would fit together without gaps. Extrusions were cut in the control surfaces to accommodate the shaft collars at the root and tip. This process included repeatedly installing the hinge rod into the hinge blocks and attaching it to the wing to ensure the control surface was at the proper distance from the wing and the dihedral remained consistent. The angle of rotation was also tested repeatedly to make sure the control surfaces could travel the required distance when installed. When these requirements were met, the hinge rod casings and shaft collars were epoxied into the control surfaces. The finished product is shown in Figure 105.

Figure 105. Elevon complete with hinge rod casing and shaft collar

A flange was added around the half line edges of each control surface piece to join the halves together. Resin was added along the flanges and the two halves were pressed together while the resin cured. Once dry, the edges were sanded down and fiberglass spheres were used to fill any gaps. The completed control surfaces were fitted with the hinge rod and installed into the hinge blocks. The control surface and wing trailing edge were sanded down until the control surfaces could rotate freely around hinge rod. Finally in position, the opposite end of the hinge blocks were bolted into the wing and secured.

Control horns were added to the midpoint of the control surfaces to be attached to the servos by linkages. The elevons were connected using nylon linkages because of the geometry of the wings. These were installed so that the majority of the linkage was inside of the glider. A hole was drilled into the wing to pass the nylon linkage through to connect to the control horn on the elevon. This can be seen in Figure 106. Similarly, the body flap was connected to the servo inside of the glider using a rigid linkage that passed through a hole in the body.

Figure 106. Elevon nylon linkages

6.3.5 Instrumentation Platforms

Platforms for the internal components were created using the same fiberglass and resin mixture to conserve weight. A 3D model of the glider was used to determine the positioning of

the internal components. Fiberglass platforms were designed around this positioning. A platform designed for the autopilot was designed such that it would be fixed to the estimated center of pressure and distribute its mass evenly in the lateral direction. A platform for the parachute was designed to elevate the parachute to the necessary height and placed adjacent to the tie down, which was fixed at the midpoint of the glider both longitudinally and laterally. A removable platform was designed to house the navigation components and was placed at the nose of the glider in an attempt to balance the mass distribution. In front of the removable platform, a small platform was installed for the pinhole camera that is located in the nose. Figures 107 – 109 describe the design and location of the instrumentation platforms.

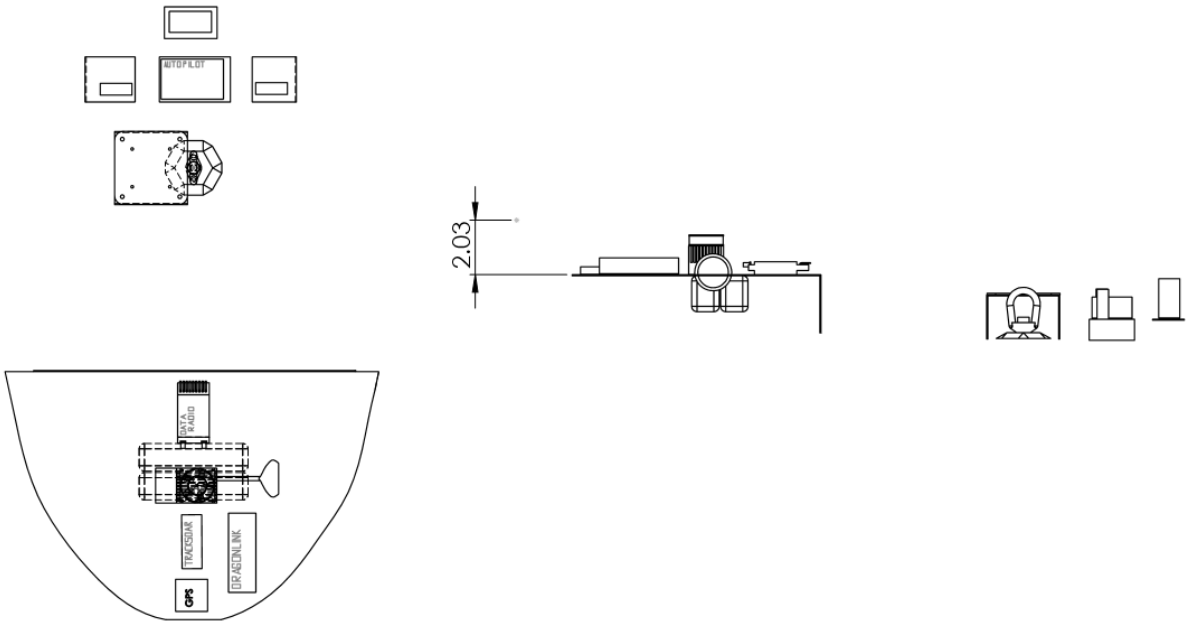


Figure 107. SolidWorks drawing of the internal components

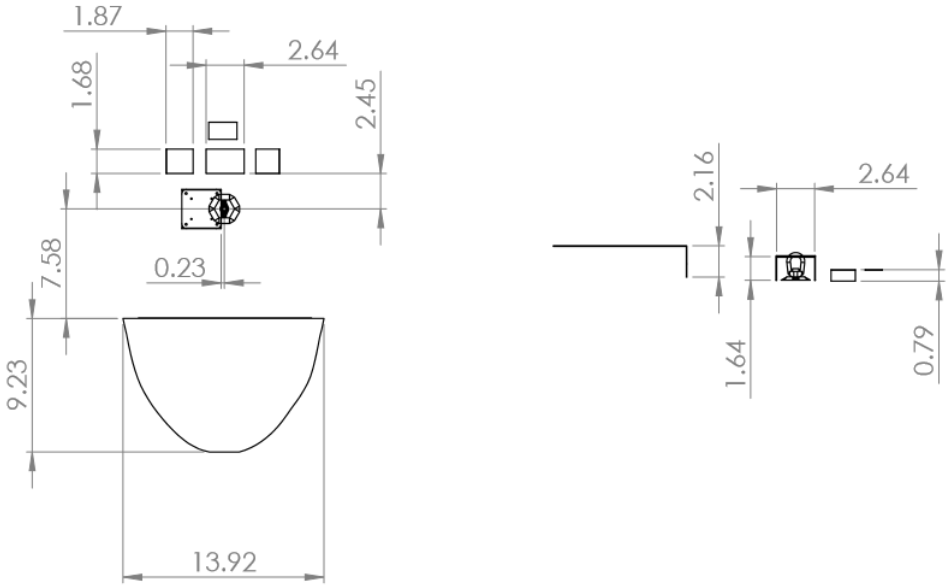


Figure 108. Dimensions of internal components


Figure 109. Sketch of internal component platforms in glider

6.3.6 Door Installation

The door was installed using quarter turn bolts that fastened the door to the body. In the body was a depressed flange that the door would fit into such that the surface of the body was as smooth as possible. Both the door and the depressed flange in the body were sanded down to make this happen. A hole was cut out of the door above the parachute to allow it to deploy. The

section that was cut out was retained and kept to fasten above the parachute in order to maintain a continuous surface. Figure 110 shows how the door was bolted into the body of the glider.

Figure 110. Door bolted into glider

6.3.7 Painting the Glider

Once the fabrication was complete, the glider was painted with gray spray paint. Each control surface was painted individually, with care taken to not plug up any holes where fasteners would be. Painting the rest of the glider was more challenging. The glider, minus the control surfaces and door, was hoisted by wire and hung from a cross beam. Spray paint was applied manually at an appropriate distance such that the paint sprayed evenly. Coats of paint were applied until the fiberglass could no longer be seen. The color gray was chosen so the glider could be seen against a blue or white sky while flying. Figure 111 shows the painted glider hanging from the cross beam.

Figure 111. Glider during the painting process

CHAPTER 7. EXPERIMENTATION

7.1 Glider Test Assembly

The glider was designed to accommodate a navigation system which would be placed on the instrumentation platforms. The removable platform would house the majority of the components, while a solitary platform at the center of mass is meant for the autopilot. The implementation of these platforms were designed around a separately developed navigation system, which has been shown in Figure 4. However, for the purposes of initial testing, a standard RC aircraft system was installed with some additional components. The primary additions to the preliminary instrumentation configuration were cameras located at the nose, in the rear, and on the door of the glider. A servo was added to the rear of the glider body to function as the release mechanism to disconnect from the high altitude balloon. Table 5 includes the components which should remain consistent throughout flight testing.

Component	**Use**
HS-5087MH Servo	Aileron/elevator
S9470SV Servo	Body flap
HS-7950TH Servo	Release mechanism
Square camera	Reference cameras
Pinhole camera	Nose camera

Table 5. Standard configuration internal components

7.2 Flight Test 1

For the initial fight test, the primary objective was to test the release mechanism and determine if the vehicle could be controlled. Due to the absence of a rudder, being able to perform a pull up maneuver was of more interest than executing a stable turn. The navigation

components installed for the first test included GPS, a Tracksoar APRS tracker, and a DVR to transmit the video from the nose camera. A FlySky FS-iA10 receiver was secured inside of the glider to be used with the FlySky FS-i10 radio controller. Figure 112 is a SolidWorks model of the instrumentation for the first flight test.

Figure 112. SolidWorks model of instrumentation for first flight test

The flight recorder consisted of a MPU6050 IMU, an Adafruit micro-SD breakout board, and an Arduino Uno. Wires were from the receiver were passed through the Arduino before terminating at the servos. A code was written on the Arduino to calculate the pulse width modulation (PWM) of the signals sent to each of the servos in an attempt to determine the servo positions during the flight. The IMU and PWM data would be recorded on an SD card using the micro-SD breakout board. The flight recorder system is shown in Figure 113.

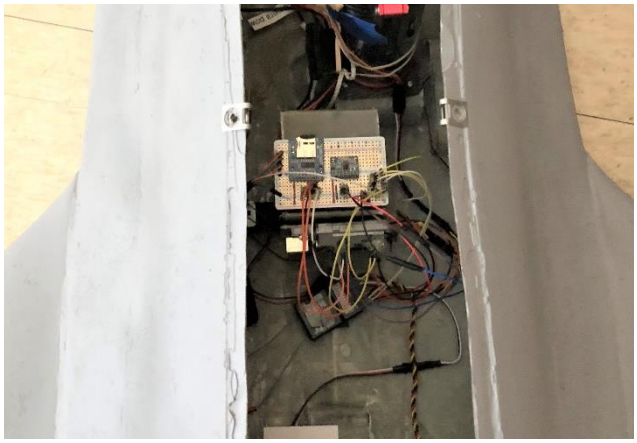


Figure 113. Data logging system for first flight test

7.2.1 Methodology

The first flight test involved launch from a tethered high altitude balloon. The glider was armed on the ground, a process which involved turning on the RC receiver, powering the data logging system, and turning on the cameras. Once armed, the glider was attached to a tethered high altitude balloon which was raised and lowered using a winch. A pin connecting the release mechanism and balloon tether linked the glider with the balloon as it ascended. FAA regulations restricted ascent to altitudes 400 feet and above, so the balloon and glider were lifted to 400 feet. At 400 feet, the balloon stopped ascending and remained tethered. Figures 114 and 115 show the glider tethered to the balloon at the ground station and at altitude, respectively.

Figure 114. Tethered glider at ground station

Figure 115. Tethered glider at 400 ft. altitude

Once the area had been cleared of non-essential personnel, the RC transmitter was powered on. Flight was initiated once a signal was sent to a servo to release the glider from the balloon. Once within 10 – 20 feet off the ground, a signal was sent to a servo to release the parachute. Upon landing, the glider was inspected for damage and the internal components were checked to ensure all connections and fixtures were intact. After landing, the balloon was retrieved using the winch.

7.2.2 Flight Conditions and Vehicle Performance

The first flight test was conducted on January 12, 2019 at the UND Observatory near Emerado, North Dakota. The temperature was recorded to be 23 °F with wind speeds 10 mph or less. Two flights were conducted at this time, one after the other. During the first flight, the glider was released from the balloon and Dr. de Leon was able to perform a pull up maneuver within a few seconds after release. Once the pull up maneuver had been completed however, the glider entered a flat spin and stability could not be recovered. The time of flight was much faster than anticipated. Preliminary calculations had estimated a time of flight of about 6 seconds, and at the time it was thought that this value was simply the time of flight for a falling object rather than an aerodynamic vehicle. Unfortunately, it was difficult to stabilize the glider. Had it been stabilized, the time of flight may have increased. The time of flight was about 8 – 9 seconds.

This was a problem because the parachute did not have enough time to fully deploy. An attempt to deploy the parachute was unsuccessful, causing the glider to crash into the ground. Upon inspection of the condition of the glider at the landing site, the parachute could be seen outside of its canister but remained tightly wound. Although this first landing caused concern for the structure of the glider, it remained relatively intact due to it landing flat. Unfortunately, no video was captured of the first flight. Figure 116 shows the damage caused by the crash landing after the first flight. Small cracks in the paint and putty used to smooth over the wing and body joint can be seen.

Figure 116. Damage from first flight crash landing

The glider was flown a second time, after it was determined that the structure was intact. This time, the glider was lifted to an altitude of 450 feet. The idea was to allow for more time to free fall before attempting a pull up maneuver in order to avoid a flat spin. This did not work out as planned. Without control input after release from the balloon, the glider entered an uncontrolled roll. Several stabilization attempts were made by executing pull up maneuvers. The glider would pitch up to a stable flight path angle only to pitch back down and enter an uncontrolled roll once again. Figure shows an uncontrolled roll sequence from the second flight during the January flight testing.

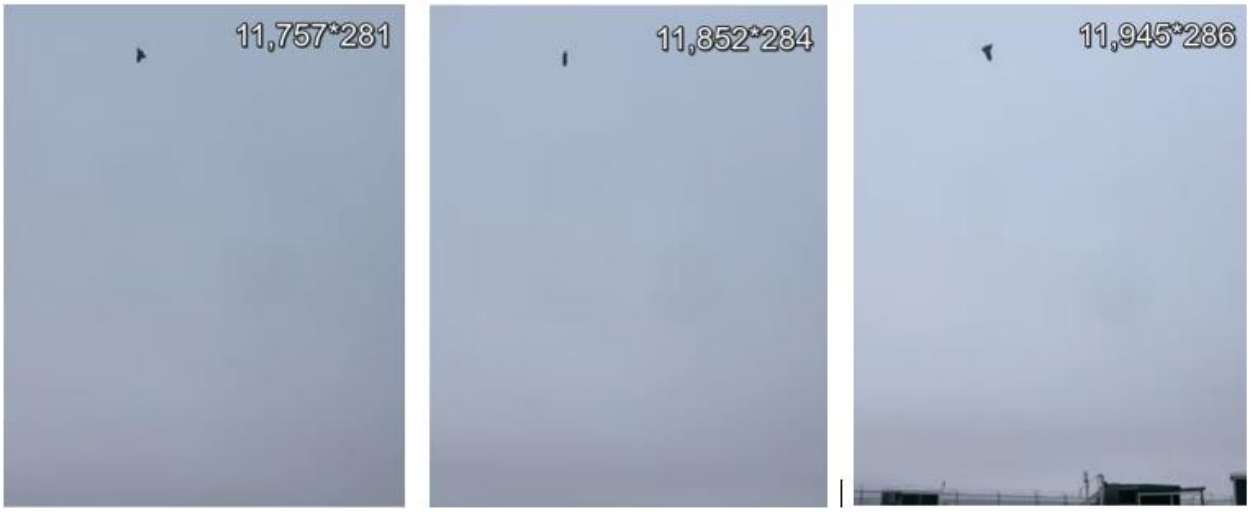


Figure 117. Uncontrolled roll sequence during second flight during January flight test

Dr. de Leon attempted to stabilize the glider before landing so it would land flat and cause minimal damage. The glider did respond to controls and stabilized a few feet off the ground, but unfortunately crashed into a barbed wire fence. The damage to the glider after it crashed into the fence was much more substantial and no more flights were attempted. Figures 118 – blank show the damage caused by the crash into the barbed wire fence.


Figure 118. Broken linkages on elevon and body flap

Figure 119. Broken release mechanism housing

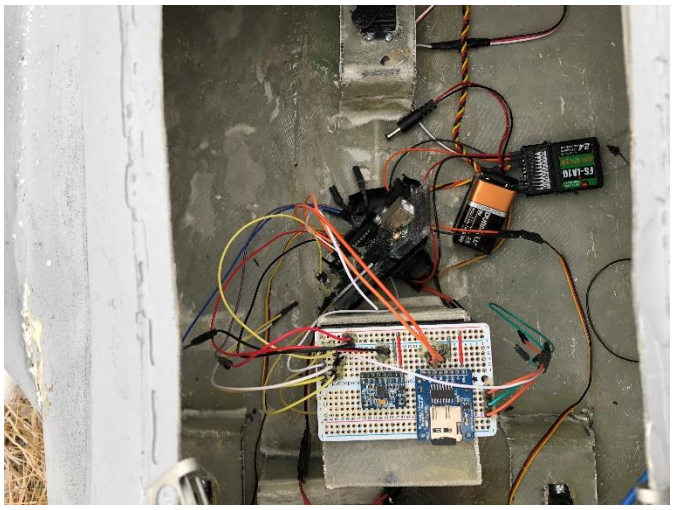


Figure 120. Damage to datalogging system

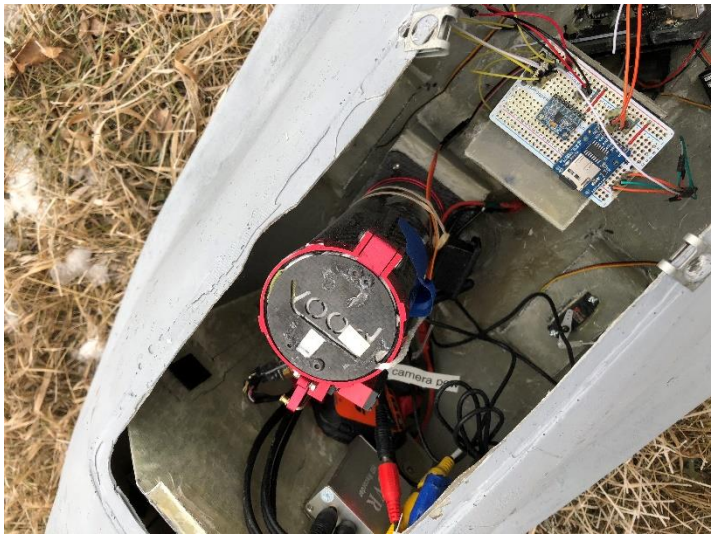


Figure 121. Removable platform displacement

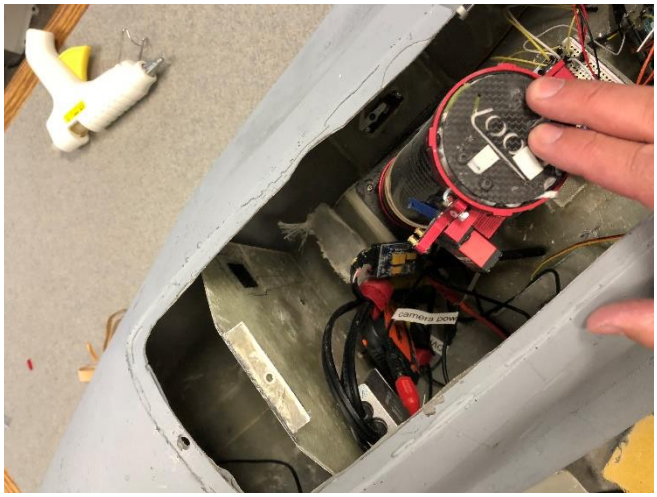


Figure 122. Parachute platform damage

Figure 123. Broken battery housing

Figure 124. Damage to wing body joint

It was known going into the first round of flight testing that the lack of a center fin and rudder was going to cause stability and control issues. However, the testing was considered successful because the glider was able to be controlled and proved to be longitudinally stable. It was determined that the servo release mechanism was sufficient because it effectively removed the glider from the balloon chord without issue. Additionally, the instability in roll was discovered using the Simulink/FlightGear flight simulation. When the glider was left unpiloted, it entered an uncontrolled roll.

An important issue was discovered during the test flights as well. Dr. Casler suggested the center of mass was too far aft. A mass balance was conducted when back in the lab and the center of mass was found to be more than 10 centimeters behind what was calculated from the Solidworks model. This was a huge stability issue because the center of mass and center of pressure were essentially in the same location.

7.2.3 Data Analysis

The data were retrieved from the SD card and imported to excel. Recorded on the SD card were readings from the IMU accelerometer, gyroscope, and magnetometer, in addition to the PWM signals for the body flap and elevons servos. The sampling rate was 10 Hz or 10 samples per second. Raw data alone cannot give much information about the aerodynamic properties of the vehicle. Various analysis techniques were used to convert the raw data into a usable format. A snapshot of the raw data is shown in Figure 125.

```
Accel(g): 0.05 0.02 1.23 Gyro(dps): -3.11 -0.85 0.37 Mag(uT): 0.00 0.00 0.00 Time(ms): 730 PWM1(ms): 1380 PWM2(ms): 292
PWM1(ms): 1380 PWM2(ms): 292 count: 760452
Accel(g): 0.05 0.01 1.23 Gyro(dps): -4.27 -1.04 0.91 Mag(uT): 0.00 0.00 0.00 Time(ms): 830 PWM1(ms): 1380 PWM2(ms): 292
PWM1(ms): 1380 PWM2(ms): 292 count: 878156
Accel(g): 0.05 0.01 1.22 Gyro(dps): -4.21 -1.04 1.16 Mag(uT): 0.00 0.00 0.00 Time(ms): 956 PWM1(ms): 1380 PWM2(ms): 292
PWM1(ms): 324 PWM2(ms): 1672 count: 1065660
Accel(g): 0.05 0.01 1.23 Gyro(dps): -3.41 -0.85 0.43 Mag(uT): 0.00 0.00 0.00 Time(ms): 1083 PWM1(ms): 1364 PWM2(ms): 1668
PWM1(ms): 1376 PWM2(ms): 1660 count: 1105524
Accel(g): 0.05 0.01 1.22 Gyro(dps): -4.09 -0.98 0.91 Mag(uT): 0.00 0.00 0.00 Time(ms): 1129 PWM1(ms): 1380 PWM2(ms): 1656
PWM1(ms): 1380 PWM2(ms): 1660 count: 1149600
Accel(g): 0.05 0.01 1.22 Gyro(dps): -4.33 -1.04 1.04 Mag(uT): 0.00 0.00 0.00 Time(ms): 1229 PWM1(ms): 1364 PWM2(ms): 280
PWM1(ms): 1364 PWM2(ms): 292 count: 1249724
Accel(g): 0.05 0.01 1.23 Gyro(dps): -3.60 -0.91 0.43 Mag(uT): 0.00 0.00 0.00 Time(ms): 1330 PWM1(ms): 1364 PWM2(ms): 292
PWM1(ms): 1364 PWM2(ms): 296 count: 1351640
Accel(g): 0.05 0.01 1.23 Gyro(dps): -3.35 -0.85 0.18 Mag(uT): 0.00 0.00 0.00 Time(ms): 1429 PWM1(ms): 1364 PWM2(ms): 292
PWM1(ms): 1364 PWM2(ms): 288 count: 1449432
```

Figure 125. Raw flight test data

7.2.3.1 Estimating Orientation Through Sensor Fusion

Sensor fusion is commonly used to combine data from more than one sensor to obtain a more reliable state estimation than what can be determined from the individual sensors. In this case, the IMU actually consists of three sensors: an accelerometer, gyroscope, and magnetometer. The accelerometer records raw acceleration measurements along each x, y, and z axis in magnitudes of g, or 9.81 m/s^2. A gyroscope records the angular rate of change in each axis, also known as the angular rates p, q, and r. The magnetometer records the strength of the Earth's magnetic field and is used to determine heading. Unfortunately, the magnetometer was not working and was not included in the sensor fusion; causing a high degree of uncertainty of the orientation in yaw. The IMU was intentionally placed in the glider so its x, y, and z axes would align with the body axes. It was mounted to the autopilot platform which is located at the estimated equilibrium point, negating the need to transfer the data from the instrument axes to the body axes.

A Kalman filter was used to fuse the raw acceleration and gyro readings to obtain estimates of the orientation and angular velocity over time. Within MATLAB, the imufilter System object can be used to fuse accelerometer and gyroscope sensor data in order to estimate device orientation [20]. The standard Kalman filter models the state vector process, $\boldsymbol{x}$, at sample time k as linear and recursive, where $\boldsymbol{A}_k$ is the state transition model and $\boldsymbol{w}_k$ is the noise vector.

$$\boldsymbol{x}_k = \boldsymbol{A}_k \boldsymbol{x}_{k-1} + \boldsymbol{w}_k \tag{73}$$

The state process $\boldsymbol{x}_k$ is not directly measurable and must be estimated by the measured state $\boldsymbol{z}_k$, where Ck is the measurement matrix relating $\boldsymbol{x}_k$ to $\boldsymbol{z}_k$ with and $\boldsymbol{v}_k$ is the noise vector.

$$\boldsymbol{z}_k = \boldsymbol{C}_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{74}$$

The noise vectors are assumed to be zero mean white processes, where E[] is the expectation operator, and are represented in a symmetric covariance matrix, $\boldsymbol{Q}$.

$$E[\boldsymbol{w}_k] = \boldsymbol{0} \tag{75}$$

$$E[\boldsymbol{v}_k] = \boldsymbol{0} \tag{76}$$

$$cov\{\boldsymbol{w}_k, \boldsymbol{w}_j\} = E[\boldsymbol{w}_k \boldsymbol{w}_j^T] = \boldsymbol{Q}_{w,k}\delta_{kj} \tag{77}$$

$$cov\{\boldsymbol{v}_k, \boldsymbol{v}_j\} = E[\boldsymbol{v}_k \boldsymbol{v}_j^T] = \boldsymbol{Q}_{v,k}\delta_{kj} \tag{78}$$

$$\boldsymbol{Q}_{w,k}{}^T = \{E[\boldsymbol{w}_k \boldsymbol{w}_k^T]\}^T = E[(\boldsymbol{w}_k \boldsymbol{w}_k^T)^T] = E[\boldsymbol{w}_k \boldsymbol{w}_j^T] = \boldsymbol{Q}_{w,k} \tag{79}$$

Kalman filters compute an unbiased *a posterori*, meaning "from the latter", estimate $\hat{\boldsymbol{x}}_k^+$ of the underlying state process $\boldsymbol{x}_k$ from extrapolation of the previous *a posterori* estimate $\hat{\boldsymbol{x}}_{k-1}^+$and the current measurement $\boldsymbol{z}_k$. The standard Kalman equations are shown in Equations 80 – 84 [18]. A linear prediction *a priori*, meaning "from the earlier", estimate $\hat{\boldsymbol{x}}_k^-$ is made by applying the linear prediction matrix $\boldsymbol{A}_k$ to the previous estimate $\hat{\boldsymbol{x}}_{k-1}^+$.

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}_k \hat{\boldsymbol{x}}_{k-1}^+ \tag{80}$$

The linearly extrapolated *a priori* error covariance matrix $\boldsymbol{P}_k^-$ is updated using the state model matrix $\boldsymbol{A}_k$ and the noise matrix $\boldsymbol{Q}_{w,k}$.

$$\boldsymbol{P}_k^- = \boldsymbol{A}_k \boldsymbol{P}_{k-1}^+ \boldsymbol{A}_k^T + \boldsymbol{Q}_{w,k} \tag{81}$$

The gain matrix $\boldsymbol{K}_k$ is updated.

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}_k^T (\boldsymbol{C}_k \boldsymbol{P}_k^- \boldsymbol{C}_k^T + \boldsymbol{Q}_{v,k})^{-1} \tag{82}$$

The *a posteriori* estimate $\hat{\boldsymbol{x}}_k^+$ is computed from the current *a priori* estimate $\hat{\boldsymbol{x}}_k^-$ and the current measurement $\boldsymbol{z}_k$.

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{C}_k \hat{\boldsymbol{x}}_k^-) \tag{83}$$

The *a posteriori* error covariance matrix $\boldsymbol{P}_k^+$ is updated.

$$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}_k) \boldsymbol{P}_k^- \tag{84}$$

MATLAB uses a complementary indirect Kalman filter to fuse the gyroscope and accelerometer data. The algorithm uses the structure described in [19] and attempts to track the errors in orientation, gyroscope offset, and linear acceleration to solve for the final orientation and angular velocity. The indirect Kalman filter models the error process, $\boldsymbol{x}_k$, with a recursive update instead of tracking orientation directly. Here, $\boldsymbol{x}_k$ is a 9-by-1 vector consisting of

$$\boldsymbol{x}_k = \boldsymbol{A}_k \boldsymbol{x}_{k-1} + \boldsymbol{w}_k \Longrightarrow \boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\theta}_k \\ \boldsymbol{b}_k \\ \boldsymbol{a}_k \end{bmatrix} = \boldsymbol{A}_k \begin{bmatrix} \boldsymbol{\theta}_{k-1} \\ \boldsymbol{b}_{k-1} \\ \boldsymbol{a}_{k-1} \end{bmatrix} + \boldsymbol{w}_k \tag{85}$$

$\boldsymbol{\theta}_k$, a 3-by-1 orientation error vector, $\boldsymbol{b}_k$, a 3-by-1 gyroscope zero rate offset vector, and $\boldsymbol{a}_k$, a 3-by-1 acceleration error vector. MATLAB corrects the components of the *a posteriori* vector $\hat{\boldsymbol{x}}_k^+$ by applying the *a posteriori* error vector. Therefore, the a priori estimate is zero and $\boldsymbol{A}_k$ is zero [18]. The MATLAB algorithm reduces the standard Kalman equations to the following:

$$\boldsymbol{x}_k^- = 0 \tag{86}$$

$$\boldsymbol{P}_k^- = \boldsymbol{Q}_{w,k} \tag{87}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}_k^T (\boldsymbol{C}_k \boldsymbol{P}_k^- \boldsymbol{C}_k^T + \boldsymbol{Q}_{v,k})^{-1} \tag{88}$$

$$\hat{\boldsymbol{x}}_k^+ = \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{C}_k \hat{\boldsymbol{x}}_k^-) \tag{89}$$

$$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}_k) \boldsymbol{P}_k^- \tag{90}$$

The MATLAB algorithm also contains innovation, or pre-fit residual, $\boldsymbol{y}_k$, and innovation covariance, $\boldsymbol{S}_k$, equations. Figure 126 and the following explanation represent the Kalman filter algorithm used in MATLAB [20].

$$\boldsymbol{y}_k = \boldsymbol{z}_k - \boldsymbol{C}_k \boldsymbol{x}_k^- = \boldsymbol{z}_k \tag{91}$$

$$\boldsymbol{S}_k = \boldsymbol{R}_k + \boldsymbol{C}_k \boldsymbol{P}_k^- \boldsymbol{C}_k^T \tag{92}$$

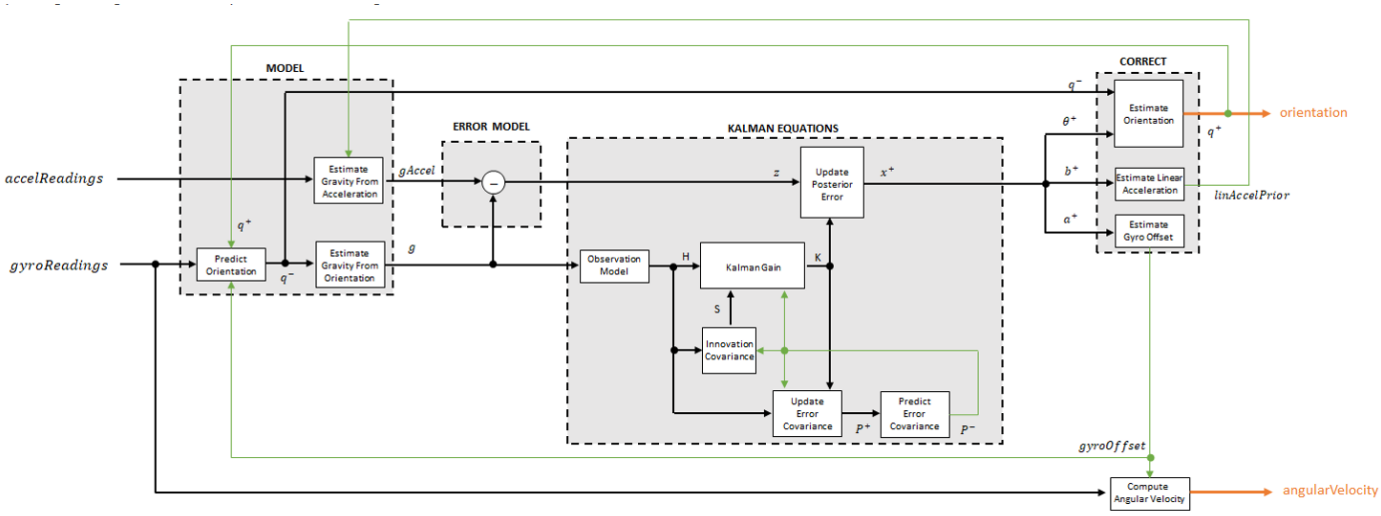


Figure 126. MATLAB Kalman filter algorithm process [20]

The MATLAB algorithm contains four blocks; the Model, Error Model, Kalman Equations, and Correct blocks. The Model block consists of the orientation prediction and gravity estimate. The orientation is predicted by estimating the angular change from the previous value, where N is the decimation factor and fs is the sample rate, shown in Equation 93. The result is converted into quaternions as shown in Equation 94. The previous estimate is then updated by rotating it by $\Delta\boldsymbol{Q}$. For the first update, the orientation estimate $\boldsymbol{q}^{-}$ is initialized assuming that the x-axis points north, shown in Equation 95.

$$\Delta\boldsymbol{\varphi}_{N\times 3} = \frac{(gyroReadings_{N\times 3} - gyroOffset_{1\times 3})}{fs} \tag{93}$$

$$\Delta\boldsymbol{Q}_{N\times 1} = quaternion(\Delta\boldsymbol{\varphi}_{N\times 3}, 'rotvec') \tag{94}$$

$$\boldsymbol{q}^{-}_{1\times 1} = (\boldsymbol{q}^{+}_{1\times 1})\left(\prod_{n=1}^{N}\Delta\boldsymbol{Q}_n\right) \tag{95}$$

An estimate of the gravity vector is made from the orientation estimate and is the transpose of the third column of $\boldsymbol{q}^{-}$. A redundant estimation of gravity is made from acceleration, shown in Equation 96.

$$gAccel_{1\times 3} = accelReadings_{1\times 3} - linAccelprior_{1\times 3} \tag{96}$$

The Error Model block computes the difference between the gravity estimate in Equation 96 and the estimate from the gyroscope readings, shown in Equation 97.

$$z = g - gAccel \tag{97}$$

The Kalman Equations block contains the equations used to output an *a posteriori* estimate, $\boldsymbol{x}^+$. The gravity estimate from the gyroscope, $\boldsymbol{g}$, and the observation of the error process, $\boldsymbol{z}$, are used to update the Kalman gain and covariance matrices. The Kalman gain is applied to an error signal, $\boldsymbol{z}$, to output $\boldsymbol{x}^+$. Within the observation model, the observed state $\boldsymbol{g}_{1x3}$ is mapped into the true state $\boldsymbol{C}_{3x9}$. The observation model is constructed as shown in Equation 98, where κ is a constant determined by the decimation factor and sample rate.

$$\boldsymbol{C}_{3\times9} = \begin{bmatrix} 0 & g_z & -g_y & 0 & -\kappa g_z & \kappa g_y & 1 & 0 & 0 \\ -g_z & 0 & g_x & \kappa g_z & 0 & -\kappa g_x & 0 & 1 & 0 \\ g_y & -g_x & 0 & -\kappa g_y & \kappa g_x & 0 & 0 & 0 & 1 \end{bmatrix} \tag{98}$$

The innovation covariance in the Kalman Equations block is used to track the variability in the measurements and is calculated from Equation 99. Here, the covariance of the observation model $\boldsymbol{R}$ is calculated using Equation 100 where κ is (decimation factor/sample rate)^2, β is the gyroscope drift noise, η is the gyroscope noise, λ is the accelerometer noise, and ξ is the linear acceleration noise.

$$\boldsymbol{S}_{3\times3} = \boldsymbol{R}_{3\times3} + (\boldsymbol{C}_{3\times9})(\boldsymbol{P}^-_{9\times9})(\boldsymbol{C}_{3\times9})^T \tag{99}$$

$$\boldsymbol{R}_{3\times3} = \big(\lambda + \xi + \kappa(\beta + \eta)\big)\boldsymbol{I} \tag{100}$$

The error estimate covariance matrix is updated as shown in Equation 101.

$$\boldsymbol{P}^+_{9\times9} = \boldsymbol{P}^-_{9\times9} - (\boldsymbol{K}_{9\times3})(\boldsymbol{C}_{3\times9})(\boldsymbol{P}^-_{9\times9}) \tag{101}$$

The a priori error estimate covariance, $\boldsymbol{P}^-$, is set to the value of $\boldsymbol{Q}$ determined in the previous iteration. The process noise covariance of the current iteration, $\boldsymbol{Q}$, is predicted as a function of the a posteriori error estimate covariance, $\boldsymbol{P}^+$. Below, v is the linear acceleration decay factor.

$$\boldsymbol{Q} = \begin{bmatrix} P^+(1) + \kappa^2 P^+(31) + \beta + \eta & 0 & 0 \\ 0 & P^+(11) + \kappa^2 P^+(41) + \beta + \eta & 0 \\ 0 & 0 & P^+(21) + \kappa^2 P^+(51) + \beta + \eta \\ -\kappa(P^+(31) + \beta) & 0 & 0 \\ 0 & P^+(41) + \beta & 0 \\ 0 & 0 & P^+(51) + \beta \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -\kappa(P^+(31) + \beta) & 0 & 0 & 0 & 0 & 0 \\ 0 & P^+(41) + \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & P^+(51) + \beta & 0 & 0 & 0 \\ P^+(31) + \beta & 0 & 0 & 0 & 0 & 0 \\ 0 & P^+(41) + \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & P^+(51) + \beta & 0 & 0 & 0 \\ 0 & 0 & 0 & v^2 P^+(61) + \xi & 0 & 0 \\ 0 & 0 & 0 & 0 & v^2 P^+(71) + \xi & 0 \\ 0 & 0 & 0 & 0 & 0 & v^2 P^+(81) + \xi \end{bmatrix}$$

The Kalman gain is computed as Equation 102 and the posterior error estimate is calculated from the Kalman gain and the gravity vector estimation errors shown in Equation 103.

$$\boldsymbol{K}_{9\times3} = (\boldsymbol{P}^-_{9\times9})(\boldsymbol{C}_{3\times9})((\boldsymbol{S}_{3\times3})^T)^{-1} \tag{102}$$

$$\boldsymbol{x}_{9\times1} = (\boldsymbol{K}_{9\times3})(\boldsymbol{z}_{1\times3})^T \tag{103}$$

The Correct block estimates the orientation by multiplying the previous estimation by the error, shown in Equation 104. The linear acceleration is updated as shown in Equation 105 and the gyroscope offset is estimated using Equation 106. Finally, the angular velocity is updated using Equation 107.

$$\boldsymbol{q}^+ = \boldsymbol{q}^-\boldsymbol{\theta}^+ \tag{104}$$

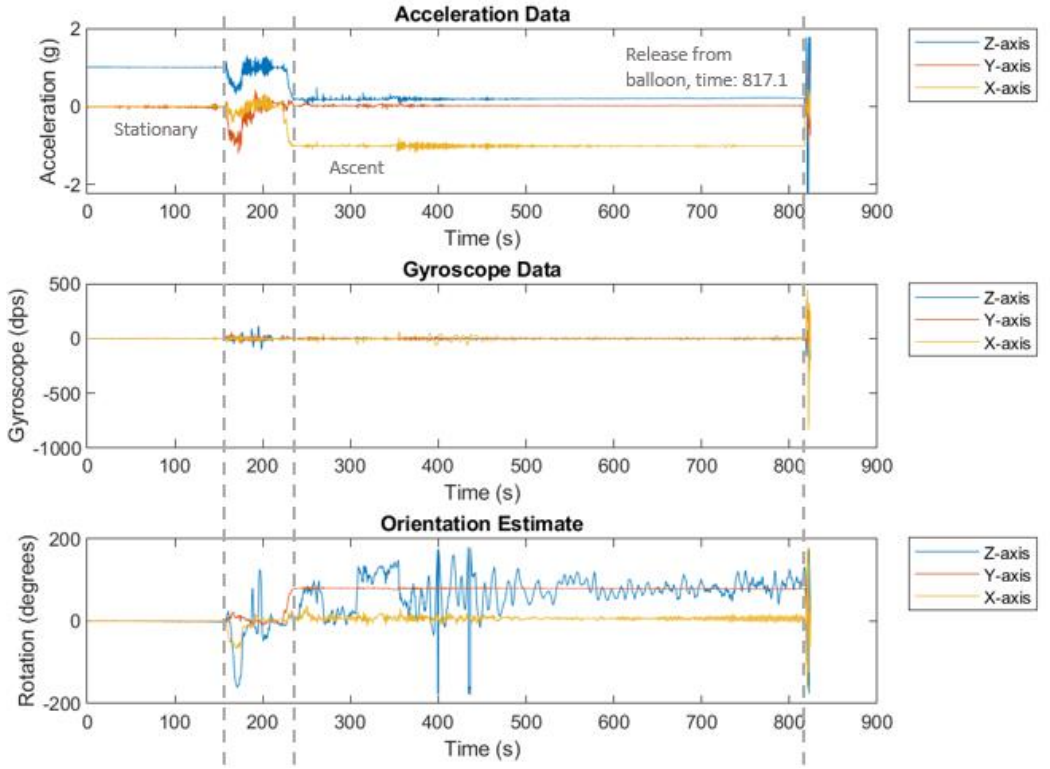
$$linAccelPrior = (linAccelPrior_{k-1})\boldsymbol{w} - \boldsymbol{a}^+ \tag{105}$$

$$gyroOffset = gyroOffset_{k-1} - \boldsymbol{b}^+ \tag{106}$$

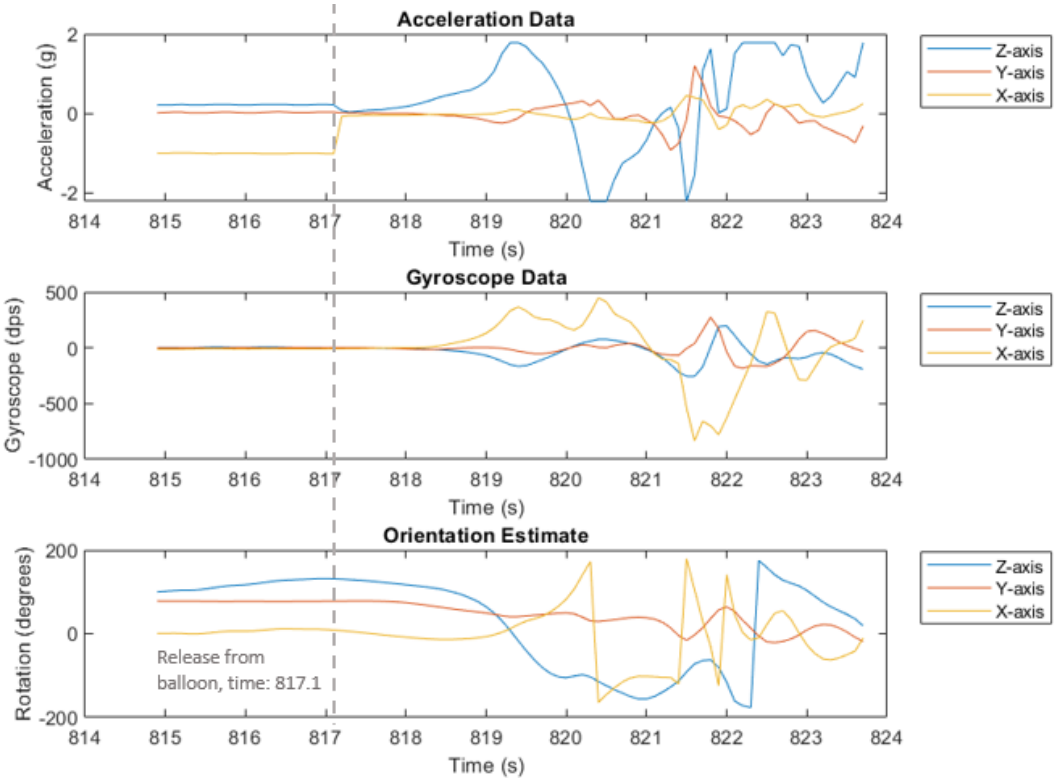
$$angularVelocity_{1\times3} = \frac{\sum gyroReadings_{N\times3}}{N} - gyroOffset_{1\times3} \tag{107}$$

The MATLAB algorithm above was used with the flight data retrieved from the January flight tests to fuse the IMU data and estimate the orientation. Figure 127 shows the raw acceleration

and gyro data for the second flight along with the estimated orientation.



(a) Glider arming, ascent, and flight



(b) Flight

Figure 127. Accelerometer and gyroscope data plotted with orientation estimate

7.2.3.2 Filtering the Data

The orientation estimate can be used to determine what angle of attack, α, and sideslip, β, to implement in the computation model for data verification. Incidence and sideslip can be found using a variety of methods but are most commonly calculated from the components of the velocity vector, shown in Equations 108 and 109.

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right) \tag{108}$$

$$\beta = \sin^{-1}\left(\frac{v}{V}\right) \tag{109}$$

Unfortunately, the glider was not equipped with an airspeed indicator during the January testing. Instead, the incidence and sideslip was reconstructed from the Euler angles. The data were imported into Simulink and first converted from rotation angles to a direction cosines matrix before being converted to incidence and sideslip angles. The results of the forces calculated using the computational model and estimated incidence and sideslip are plotted against the estimated forces calculated from the accelerometer of the IMU in Figure 128.

It can be seen from the figure that the two sets of data do not match. The biggest contrast is the smoothness of the IMU data compared to the irregular computational data. Of biggest concern is the significant changes in the computational data over very small time intervals. Because the IMU data did not exhibit similar trends, it is thought that the error was within the reconstruction of the flight rather than the raw data itself. Further investigation revealed the primary issue was with the estimation of the incidence and sideslip angles. Without raw data to compare, the actual incidence and sideslip angle is essentially unknown because the error cannot be determined.

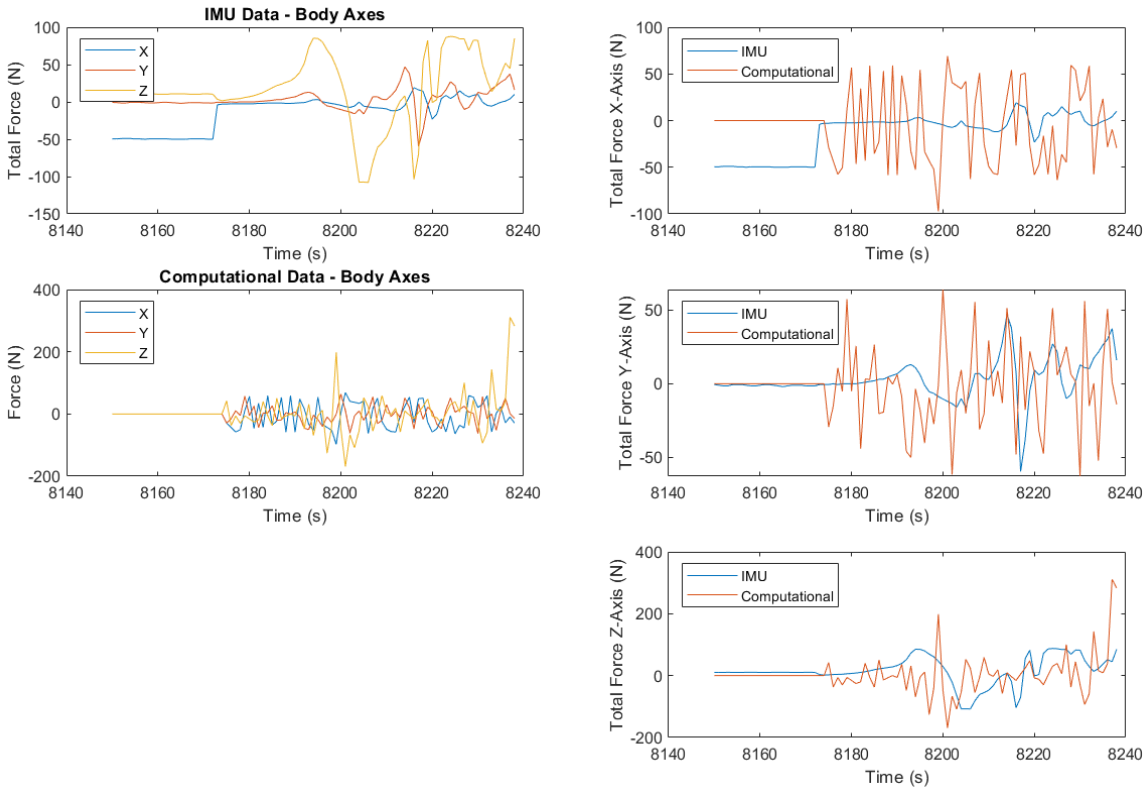


Figure 128. IMU vs. Computational Model Aerodynamic Forces

A second Kalman filter was implemented in order to get a better estimation of the incidence and sideslip angles. An extended Kalman filter was used in order to estimate the state of a discrete-time nonlinear system. This type of filter was chosen because of the instability of the second flight. A traditional Kalman filter relies on the system and measurement functions to be linear with respect to the state variables, which may not be the case with this type of vehicle and the unstable behavior observed during the flight. The state representation of the system model and measurement model is shown in Equations 110 and 111 [20]. The state and measurement models are shown in Equation 113 and Equation 114, respectively. The noise vectors $\boldsymbol{w}$ and $\boldsymbol{v}$ were obtained using the default noise parameters for the IMU.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) + \boldsymbol{w}(t) \tag{110}$$

$$\boldsymbol{z} = \boldsymbol{H}(t)\boldsymbol{x}(t) + \boldsymbol{v}(t) \tag{111}$$

$$\begin{bmatrix} \Delta\dot{u} \\ \Delta\dot{v} \\ \Delta\dot{w} \\ \Delta\dot{\varphi} \\ \Delta\dot{\theta} \\ \Delta\dot{q} \\ \Delta\dot{h} \\ \Delta\dot{p} \\ \Delta\dot{r} \\ \Delta\dot{\psi} \end{bmatrix} = \left[\begin{array}{cccccccccc} X_u & 0 & X_w & 0 & -g & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & g\cos\theta & 0 & -(u_0 - Y_r) & 0 & Y_p & 0 & 0 \\ Z_u & 0 & Z_w & 0 & 0 & u_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ M_u & 0 & M_w & 0 & 0 & M_q & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -u_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & L_v & 0 & 0 & 0 & 0 & 0 & L_p & L_r & 0 \\ 0 & N_v & 0 & 0 & 0 & 0 & 0 & N_p & N_r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta w \\ \Delta\varphi \\ \Delta\theta \\ \Delta q \\ \Delta h \\ \Delta p \\ \Delta r \\ \Delta\psi \end{bmatrix} + [0][0] + [0.00008] \tag{113}$$

$$\begin{bmatrix} \Delta\dot{u} \\ \Delta\dot{v} \\ \Delta\dot{w} \\ \Delta\dot{\varphi} \\ \Delta\dot{\theta} \\ \Delta\dot{q} \\ \Delta\dot{h} \\ \Delta\dot{p} \\ \Delta\dot{r} \\ \Delta\dot{\psi} \end{bmatrix} = \left[\begin{array}{cccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta w \\ \Delta\varphi \\ \Delta\theta \\ \Delta q \\ \Delta h \\ \Delta p \\ \Delta r \\ \Delta\psi \end{bmatrix} + [0][0] + \begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0001 \\ 0.0001 \\ 0.0001 \\ 0.0096 \\ 0.0096 \\ 0.0096 \\ 0.0001 \end{bmatrix} \tag{114}$$

$$X_u = \frac{-(C_{D_u} + 2C_{D_0})\bar{q}S_{ref}}{mu_0} \tag{115}$$

$$X_w = \frac{-(C_{D_a} - C_{L_0})\bar{q}S_{ref}}{mu_0} \tag{116}$$

$$Y_\beta = \frac{C_{y_\beta}\bar{q}S_{ref}}{m} \tag{117}$$

$$Y_v = Y_\beta u_0 \tag{118}$$

$$Y_p = \frac{C_{y_p}\bar{q}S_{ref}b_{ref}}{2mu_0} \tag{119}$$

$$Y_r = \frac{C_{y_r}\bar{q}S_{ref}b_{ref}}{2mu_0} \tag{120}$$

$$Z_u = \frac{-(C_{L_u} + 2C_{L_0})\bar{q}S_{ref}}{mu_0} \tag{121}$$

$$Z_w = \frac{-(C_{L_a} + 2C_{D_0})\bar{q}S_{ref}}{mu_0} \tag{122}$$

$$M_u = C_{m_u}\frac{\bar{q}S_{ref}\bar{c}}{I_y u_0} \tag{123}$$

$$M_w = C_{m_a}\frac{\bar{q}S_{ref}\bar{c}}{I_y u_0} \tag{124}$$

$$M_q = C_{m_q}\frac{\bar{c}}{2u_0}\frac{\bar{q}S_{ref}\bar{c}}{I_y} \tag{125}$$

$$L_\beta = \frac{C_{l_\beta} qSb}{I_x} \tag{126}$$

$$L_v = L_\beta u_0 \tag{127}$$

$$L_p = C_{l_p}\frac{qSb^2}{2I_x u_0} \tag{128}$$

$$L_r = C_{l_r}\frac{qSb^2}{2I_x u_0} \tag{129}$$

$$N_\beta = \frac{C_{l_\beta} qSb}{I_z} \tag{130}$$

$$N_v = N_\beta u_0 \tag{131}$$

$$N_p = C_{n_p}\frac{qSb^2}{2I_z u_0} \tag{132}$$

$$N_r = C_{n_r}\frac{qSb^2}{2I_z u_0} \tag{133}$$

The purpose of this filter was to obtain a better incidence and sideslip angle estimation from the IMU data and velocity estimates. Figure 129 displays the implementation of the Kalman filter in Simulink. A MATLAB function block creates the state and measurement matrices which are fed to an Extended Kalman Filter block containing the additive noise vectors.

The estimates are then fed to a MATLAB function which calculates the incidence and sideslip angles to be used in the computational model.

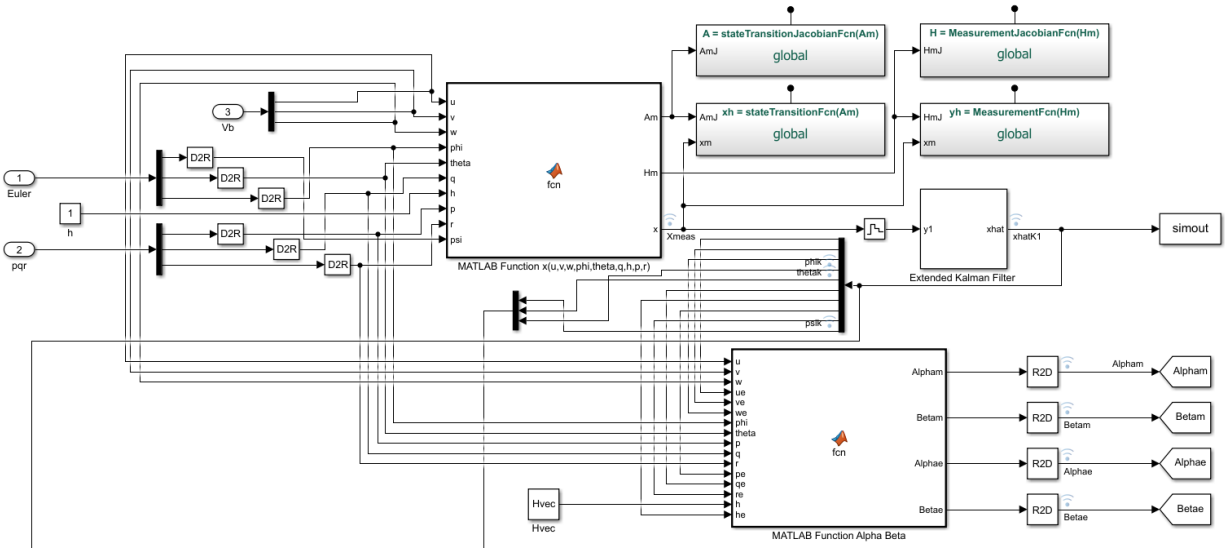


Figure 129. Kalman Filter implementation in Simulink

The results from second Kalman filter showed improvement, but there were indications that the values were still not accurate. This can be seen in Figure 130, where there is a lot of noise in the beginning of the flight test. The values smooth out and the estimate improves over time, but there is still too much variability to make a reasonable attempt at verification. Therefore, it will be necessary to conduct additional flight testing in the future with different flight hardware which could more effectively record the flight data.

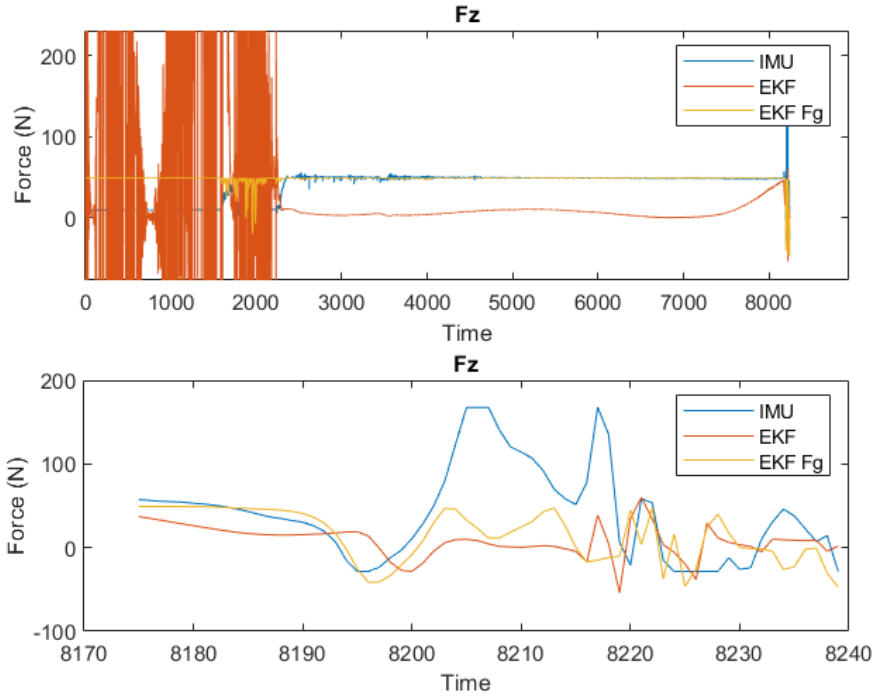


Figure 130. F_z IMU and Extended Kalman Filter (EKF) comparison

7.2.4 Comparison to Simulink/FlightGear Simulation Data

Using FlightGear to visualize the vehicle dynamics is beneficial because as flight maneuvers are executed, any signal within the Simulink environment can be recorded. This environment was utilized to perform a pull up maneuver during the flight simulation and have various signals recorded in an effort to make a comparison to the test flight data. On the first attempt, the pull up maneuver was not effective. However, the test flight proved that glider was controllable and that the body flap did induce a nose up moment. Further investigation revealed the body flap pitching moment was not high enough to redirect the vehicle dynamics from the datum aerodynamics.

At this point, the De Havilland Beaver joystick model in Simulink/FlightGear was used to get an idea of the traditional aerodynamics of the pull up maneuver [23]. While the Beaver does not use a body flap in the way the glider does, the objective was to observe the pitching moment due to elevator change. Within the Simulink environment, the force, moment, and

actuator signals were recorded and a nose up maneuver was performed using the FlightGear flight simulator to collect the data. It was observed that the pitching moment due to the actuator deflection was around three times the magnitude of the nominal pitching moment.

The Preliminary Subsonic Aerodynamic Model for the HL-20 was referenced to determine the change in the pitching moment magnitude due to body flap deflection [2]. Again, the pitching moment due to control surface deflection was of a magnitude of about two to three times greater than the nominal pitching moment. Some trial and error was used to apply a gain to the pitching moment coefficient value calculated within Simulink. With a gain of two applied, the pitching moment still was not large enough to produce the nose up moment that was observed during the flight testing. A gain of three was applied, and the glider was able to execute a nose up maneuver very similar to what was observed in the flight testing. Inspection of the recorded data revealed the results were very similar. The results are displayed in Figure 131.

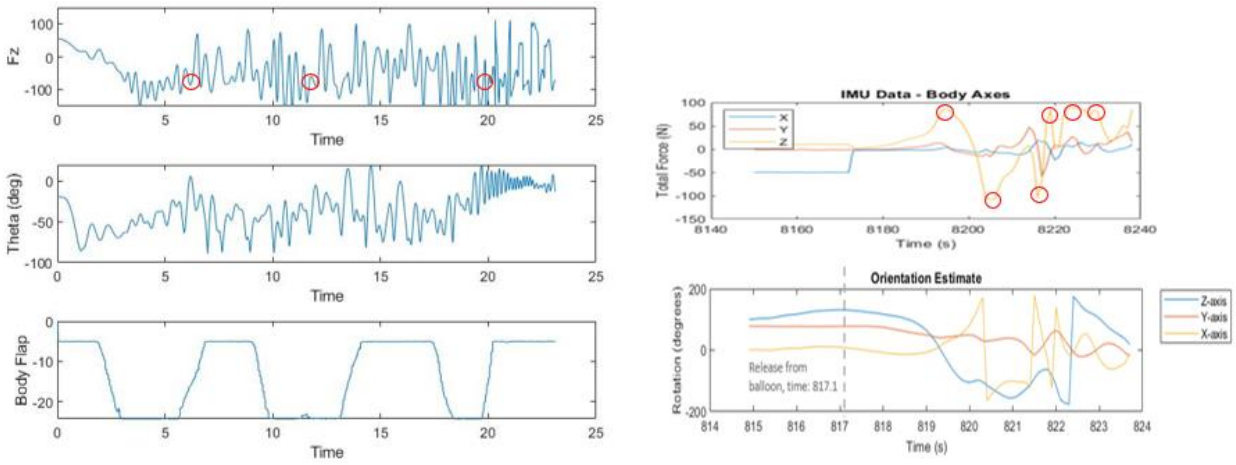


Figure 131. Recorded FlightGear data compared to flight test data for pull up maneuver

The data were examined such that points where the body flap was being deflected and the angle theta was around 0 degrees were identified and are circled in red in Figure 131. Similar points were identified within the flight test data and both were plotted against each other for

comparison. Of interest is the magnitude of the force in the z-direction, which hovered between 90 and 100 N within the flight test data. The direction of the force, positive or negative, is of less interest because the glider was periodically entering an uncontrolled roll. This caused the sensor value to record positive and negative values. The sampling of the data makes it nearly impossible to find a clear stretch where no instability was observed, so the comparison mainly looked at the magnitudes since the directions were highly irregular between both sets of data.

The magnitude of the force in the z-axis recorded in Simulink falls within the same range of 90 to 100 N when the body flap is deflected and theta is around 0 degrees. This suggests the aerodynamics are adequately modeled to estimate the vehicle dynamics, although the percent error will be unknown until more flight data are collected. It is thought that the gain applied to the body flap pitching moment coefficient indicates there are contributions in the aerodynamics that are not properly incorporated into the computational code. The errors most likely stem from the negative camber on the body flap and the blunt body of the fuselage affecting the downwash onto the body flap. A more thorough investigation into the 2D aerodynamics should be conducted using XFOIL. Additionally, the camber of the body flap must be incorporated into the 3D VLM. Preliminary work indicated the mean surface of the control surface would be adequate, as is traditionally done. However, due to the negative camber on the fuselage and the data mismatch, it is thought that the body flap should be modeled using the same method as the fuselage rather than using the traditional technique.

CHAPTER 8. DISCUSSION AND FUTURE WORK

While the aerodynamics of the glider were not verified as an outcome of the flight testing, the various methods used during the development exhibit similar results. The combination of the different data types suggest the aerodynamics may be adequately modeled, but the degree of error will be undefined until comprehensive flight testing is conducted. A positive outcome of the flight testing was that the vehicle was controllable which supports future test efforts.

Flight testing of a vehicle of this type is challenging due to restrictions imposed by the FAA where it can take months to obtain a flight waiver. Alternatively, low altitude tests which don't require FAA clearance like the one conducted in this analysis present problems of their own. A drop height of 400 feet results in a time of flight less than 10 seconds. The amount of data gathered in this timeframe is not sufficient to adequately asses the aerodynamics of a vehicle. Additionally, a short time of flight prevents the parachute from deploying properly which is a risk to the vehicle.

In the absence of sufficient flight data, further investigation of the computational model must be conducted. A second look into the issues with Solidworks Flow Simulation may be helpful to understand if any of the issues with the moment calculations and lateral forces can be mitigated in order to support future simulations. Additionally, there may be open source CFD programs that are more comprehensive and could possibly be used. However, open source programs can pose a large learning curve, such as OpenFOAM which requires a Linux platform.

While it is clear that further development of the computational model is necessary, attempting to replicate dynamics of the vehicle without supporting data is challenging. The computational model should be developed to include aerodynamic effects such as vortex

shedding. However, the unusual geometry of the vehicle make it difficult to be confident that the modeling of more complex flows is done accurately. Traditional evaluation techniques would include wind tunnel testing which could shed some light on the flow field around the vehicle and would be helpful in this case.

Future work will include more extensive flight testing of the vehicle to ascertain the aerodynamic properties over the flight envelope. Of primary concern should be the lateral stability, which has proven difficult to model computationally. Improvements to the lateral stability could be accomplished by adding a rudder or wing flaps, although the design is limited by the mass balance. Effort must be made to prevent the glider from being too tail-heavy.

A more robust navigation system will also be used to ensure the flight data will be more useful in the analysis. The new navigation system will include an airspeed sensor and a working magnetometer. These sensors will aid in the orientation, angle of attack, and sideslip estimate. Additionally, the new navigation system will be capable of reading the signals sent to the servos so the control surface deflections can be recorded. While the new navigation system is limited in range, it is believed that additional flight testing from higher altitudes within range can be conducted.

CHAPTER 9. CONCLUSIONS

A high altitude lifting body glider has been developed and flight tested. The purpose of this research is to demonstrate if a lifting body scale model is a suitable aerodynamic design that can be used as a recovery method for high altitude ballooning payloads. While the glider has not been tested at high altitudes, flight testing has demonstrated that the glider is controllable. The aerodynamics of the glider have been verified through various analysis methods, namely CFD and flight simulations, although the error with respect to the actual flight characteristics cannot definitively be determined. However, due to the qualitative evidence observed during the flight tests, there is confidence that the computational aerodynamics within the model do represent the general dynamics of the vehicle.

REFERENCES

[1] de Leon, Pablo. "New Developments in Atmospheric Reentry Vehicles". Atmospheric Reentry Vehicles and Systems. March 2001, Print.

[2] B. E. Jackson, C.I. Cruz, Preliminary Subsonic Aerodynamic Model for Simulation Studies of the HL-20 Lifting Body. 1992.

[3] MATLAB Workflow for Tuning the HL-20 Autopilot.https://www.mathworks.com/help/control/examples/HL20MatlabWorkflowExample.html, (accessed 15.9.18).

[4] Bertin, John J., and Russell M. Cummings. Aerodynamics for Engineers. 5. ed., int. ed. ed. Upper Saddle River, NJ: Pearson Prentice-Hall, 2009. Web.

[5] Melin, Tomas. "A Vortex Lattice MATLAB Implementation for Linear Aerodynamic Wing Applications." Royal Institute of Technology (KTH), 2000. Print.

[6] Anderson, John David. Fundamentals of Aerodynamics. 5. ed. New York, NY [u.a.]: McGraw-Hill, 2011.

[7] McBain, G. D. Theory of Lift: Introductory Computational Aerodynamics with MATLAB and Octave. 1st ed. New York: John Wiley & Sons, Incorporated, 2012. Aerospace Ser Web.

[8] Boyden, R. P., and D. C. Freeman Jr. Subsonic and Transonic Dynamic Stability Characteristics of a Space Shuttle Orbiter., 1975. Web.

[9] Huffman, Jarrett K., Charles H. Fox, and Bernard Spencer Jr. A Study to Determine Methods of Improving the Subsonic Performance of a Proposed Personnel Launch System (PLS) Concept., 1995. Print.

[10] Nelson, Robert C. *Flight Stability and Automatic Control*. 2. ed, internat. ed. Boston, Mass. [u.a.]: McGraw Hill, 1998. Web.

[11] "6DOF (Euler Angles)." *MathWorks.com.* Web. Mar 26, 2019 <https://www.mathworks.com/help/aeroblks/6dofeulerangles.html>.

[12] "asbFlightControlAnalysis" *MathWorks.com.* Web. Mar 31, 2019 <https://www.mathworks.com/help/aeroblks/asbflightcontrolanalysis.html>.

[13] FlightGear contributors. *FlightGear Flight Simulator*. <http://www.flightgear.org/about/features/>.

[14] Blender Online Community. *Blender - a 3D Modelling and Rendering Package*. Blender Institute, Amsterdam.< http://www.blender.org>.

[15] Caughey, David A. *Introduction to Aircraft Stability and Control Course Notes for M&AE 5070*. Sibley School of Mechanical & Aerospace Engineering Cornell University, 2011. Web.

[16] "About Aerospace Coordinate Systems" *MathWorks.com*. Web. Apr 9, 2019 <https://www.mathworks.com/help/aeroblks/about-aerospace-coordinate-systems.html>.

[17] "Working with the Flight Simulator Interface" *MathWorks.com*. Web. Apr 9, 2019 <https://www.mathworks.com/help/aeroblks/working-with-the-flight-simulator-interface.html>.

[18] Hibbeler, Russell C. *Mechanics of Materials*. 9th edition ed. Upper Saddle River, N.J: Pearson Education, 2003.

[19] "Shear Forces and Bending Moments." Web. Apr 15, 2019 <http://academic.uprm.edu/pcaceres/Courses/MMII/IMoM-4A.pdf>.

[20] "imufilter System Object" MathWorks.com. Web. Jun 2, 2019 <https://www.mathworks.com/help/fusion/ref/imufilter-system-object.html>.

[21] “Open Source Sensor Fusion”. Web. Jun 3, 2019 <https://github.com/memsindustrygroup/Open-Source-Sensor-Fusion/tree/master/docs>.

[22] Ascorti, Leonardo. “An Application of the Extended Kalman Filter to the Attitude Control of a Quadrotor.” Polytechnic University of Milan, 2013. Web.

[23] Fly the De Havilland Beaver. https://www.mathworks.com/help/aeroblks/fly-the-dehavilland-beaver.html, (accessed 26.11.19).

APPENDIX A – COMPUTATIONAL MODEL FUNDAMENTAL CODE

```
% glider_geo.m
% glider 3D panel ------------------------------------------------------
% first partition geometry
c_r1 = 958.919/1000; % root chord, m
c_t1 = 390.065/1000;
taper1 = c_t1/c_r1;
s1 = 415.476/2000; % semi-span, m
b1 = 2 * s1; %span, m
AR1 = (2 * b1) / (c_r1 + c_t1);
S1 = 2 * 0.5 * c_r1 * s1;
cmac1 = (2/3) * c_r1 * ((1 + taper1 + taper1^2) / (1 + taper1)); % ref point
sweep1 = pi/2 - atan(s1/(c_r1 - c_t1)); % LE sweep angle
dihedral1 = 0;
twist1 = 0;
nchord1 = 8;
nspan1 = 8;
S_ref = ((c_t1 + c_r1)/2) * ((b1)/2) * 2;
AR_ref = ((s1)*2)^2 / S_ref;

% body flap
c_rb = 77.811/1000; % m
sb = 378/2000; % m
bb = sb * 2;
Sb = c_rb * bb;
ARb = (bb^2)/Sb;

% meshwing 1st
 [x, y, z] = meshwing_delta_bodyflap (AR_ref, nchord1, nspan1, sweep1, 0, taper1, 0, s1, c_r1, c_t1, sb, c_rb, alphab);

% second partition
c_r2 = c_t1 - 0.096276/2; % root chord, m
c_t2 = 0.096276/2; % tip chord, m
taper2 = c_t2/c_r2;
s2 = 264.11/1000; % semi-span, m
b2 = 2 * s2; %span, m
AR2 = (2 * b2) / (c_r2 + c_t2);
S2 = 2 * 0.5 * c_r2 * s2;
cmac2 = (2/3) * c_r2 * ((1 + taper2 + taper2^2) / (1 + taper2)); % ref point
sweep2 = pi/2 - atan(s2/(c_r2 - c_t2));
dihedral2 = 17.67 *(pi/180);
twist2 = 0;
nchord2 = 6;
nspan2 = 12;

S_wref = ((c_t2 + 0.6224) / 2) * ((b1 + b2) / 2) * 2;
cmac_ref = cmac1*(S1/(S1+S2)) + cmac2*(S2/(S1+S2)) + (1/4*c_rb)*(Sb/(S1+S2));
se = s2; % semi-span, m
bf = se*2;

% meshwing 2nd
ym2 = linspace(-s2, s2, nspan2 +1);
le_x2 = (tan (sweep2) * abs (ym2)); % row vector
le_z2 = tan (dihedral2) * abs (ym2); % row vector
xi2 = linspace (0, c_r2, nchord2 +1);
meanline2 = [xi2.', camber(xi2,0,0)]; %function camber(xi)
c2 = c_r2 - le_x2;
```

A1. MATLAB script glider_geo.m

```
twists2 = 2 * twist2 * abs(ym2);
dx2 = meanline2 * ([c2; c2] .* [cos(twists2); -sin(twists2)]);
dz2 = meanline2 * ([c2; c2] .* [sin(twists2); cos(twists2)]);
dx4 = [0.0032 0.0046 0.0060 0.0074 0.0088 0.0102 0.0116 0.0102 0.0088 0.0074 0.0060 0.0046 0.0032];
xx2 = repmat (le_x2, size(xi2.')) + dx2/(c_r2+c_t2/4)+dx4;
yy2 = repmat (ym2, size(xi2.'));
zz2 = repmat (le_z2, size(xi2.')) + dz2;

X2 = zeros(nchord2+3, nspan2+1);
X2 = xx2;
x3  = X2(7,:) + (c_t2)/2;
x4 = x3 + (c_t2)/2;
j = nspan2+ 1;
   for i = 1:j
      X2(8,i) = x3(:,i);
      X2(9,i) = x4(:,i);
   end
% replacing with elevon deflection
x2(:,1) = X2(:,1);
x2(:,2) = X2(:,2);
x2(:,3) = X2(:,3);
x2(:,4) = X2(:,4);
x2(:,5) = X2(:,5);
x2(:,6) = X2(:,6);
x2(:,7) = X2(:,7);
x2(:,8) = X2(:,7);
x2(:,9) = X2(:,8);
x2(:,10) = X2(:,9);
x2(:,11) = X2(:,10);
x2(:,12) = X2(:,11);
x2(:,13) = X2(:,12);
x2(:,14) = X2(:,13);
if alphaeL ~=0
   for i = 1:7
   x2(8,i) = x2(7,i) + cos(alphaeL/2)*(c_t2/2);
   x2(9,i) = x2(8,i) + cos(alphaeL/2)*(c_t2/2);
   end
end
if alphaeR ~= 0
   for i = 7:14
      x2(8,i) = x2(7,i) + cos(alphaeR/2)*(c_t2/2);
      x2(9,i) = x2(8,i) + cos(alphaeR/2)*(c_t2/2);
   end
end

Y2 = zeros(nchord2+3, nspan2+1);
Y2 = yy2;
   for i = 1:j
      Y2(8,i) = yy2(7,i);
      Y2(9,i) = yy2(7,i);
   end

y2(:,1) = Y2(:,1) - s1;
y2(:,2) = Y2(:,2) - s1;
y2(:,3) = Y2(:,3) - s1;
y2(:,4) = Y2(:,4) - s1;
y2(:,5) = Y2(:,5) - s1;
y2(:,6) = Y2(:,6) - s1;
```

A1. MATLAB script glider_geo.m (continued)

```
y2(:,7) = Y2(:,7) - s1;
y2(:,8) = Y2(:,7) + s1;
y2(:,9) = Y2(:,8) + s1;
y2(:,10) = Y2(:,9) + s1;
y2(:,11) = Y2(:,10) + s1;
y2(:,12) = Y2(:,11) + s1;
y2(:,13) = Y2(:,12) + s1;
y2(:,14) = Y2(:,13) + s1;

Z2 = zeros(nchord2+3, nspan2+1);
Z2 = zz2;
   for i = 1:j
      Z2(8,i) = zz2(7,i);
      Z2(9,i) = zz2(7,i);
   end

z2(:,1) = Z2(:,1);
z2(:,2) = Z2(:,2);
z2(:,3) = Z2(:,3);
z2(:,4) = Z2(:,4);
z2(:,5) = Z2(:,5);
z2(:,6) = Z2(:,6);
z2(:,7) = Z2(:,7);
z2(:,8) = Z2(:,7);
z2(:,9) = Z2(:,8);
z2(:,10) = Z2(:,9);
z2(:,11) = Z2(:,10);
z2(:,12) = Z2(:,11);
z2(:,13) = Z2(:,12);
z2(:,14) = Z2(:,13);
if alphaeL ~=0
   for i = 1:7
   z2(8,i) = z2(7,i) - sin(alphaeL/2)*(c_t2/2);
   z2(9,i) = z2(8,i) - sin(alphaeL/2)*(c_t2/2);
   end
end
if alphaeR ~= 0
   for i = 7:14
      z2(8,i) = z2(7,i) - sin(alphaeR/2)*(c_t2/2);
      z2(9,i) = z2(8,i) - sin(alphaeR/2)*(c_t2/2);
   end
end
```

A1. MATLAB script glider_geo.m (concluded)

```
% function meshwing adapted from McBain 2012
function [x, y, z] = meshwing_delta_bodyflap (AR, nchord, nspan, sweep, dihedral, taper, twist, s, c_r, c_t, sb, c_rb, alphab) % no
camber
y = linspace(-s , s , nspan +1);
yb = linspace(-sb, sb, nspan+1);
le_x = tan (sweep) * abs (y); % row vector
le_z = tan (dihedral) * abs (y); % row vector
xi = linspace (0, c_r, nchord +1);
xbi = linspace(0, c_rb, 3);
manualcamber = [0, (0.1137-0.0793)/2, (0.1438-0.1018)/2, (0.1523-0.1121)/2, (0.1443-0.1137)/2, (0.1315-0.1137)/2, ...
   (0.1336-0.1129)/2, (0.1422-0.1057)/2, (0.1464-0.0921)/2];
meanline = [xi.', manualcamber']; %function camber(xi)
c = c_r - le_x; % chord length distribution for a delta wing
twists = 2 * twist * abs(y);
dx = meanline * ([c; c] .* [cos(twists); -sin(twists)]);
dz = meanline * ([c; c] .* [sin(twists); cos(twists)]);
x2 = repmat (le_x, size(xi.')) + (1/c_r)*dx; % random multiplier guess and check
y2 = repmat (y, size(xi.'));
yb = repmat (yb, size(xbi.'));
z2 = repmat (le_z, size(xi.')) + dz;

x = zeros(nchord+1, nspan+1);
x = x2;
x3  = x(9,:) + cos(alphab)*(c_rb/6);
x4 = x3 + cos(alphab)*(5*c_rb/6);
j = nspan+ 1;
   for i = 1:j
      x(10,i) = x3(:,i);
      x(11,i) = x4(:,i);
   end
y = zeros(nchord+1, nspan+1);
y = y2;
   for i = 1:j
      y(10,i) = yb(2,i);
      y(11,i) = yb(3,i);
   end
z = zeros(nchord+1, nspan+1);
z = z2;
z3 = z(9,:) - sin(alphab)*(c_rb/6);
z4 = z3 - sin(alphab)*(c_rb*(5/6));
   for i = 1:j
      z(10,i) = z3(:,i);
      z(11,i) = z4(:,i);
   end


end
```

A2. MATLAB function meshwing_delta_bodyflap.m

```
% function betaloop2
function [L_bt, D_bt, Y_bt, Mbody_Test, Mwing_Test2,Mbody_Test2, ...
Mwing_Test3,Mbody_Test3] = BetaLoop2(v, q, qnb, qr, alpha, r, r1, r2, rnb, r1nb, ...
r2nb, dY, Dwx, Dwy, Dwz, dYnb, dens, w2b, w2nb, w2, Dwx_nb, Dwy_nb, Dwz_nb, rr, ...
rr1, rr2, Dwxr, Dwyr, Dwzr, com, c_r1, c_t1, dYb_s, dYbnb_s, dYf_s, dYr, rTe, ...
Cm_xf, s1)
% beta loop ------------------------------------------------------------
i = 1;
for beta = -10:2:12
%alpha = 0;
rhs_b(:,i) = boundarycondition(v, q, alpha * (pi/180), beta * (pi/180), r, r1, r2);
gamma_s_b(:,i) = w2b \ -rhs_b(:,i);
rhsnb_b(:,i) = boundarycondition(v, qnb, alpha * (pi/180), beta * (pi/180), rnb, r1nb, r2nb);
gamma_snb_b(:,i) = w2nb \ -rhsnb_b(:,i);
i = i+1;
end
% resolving the gammas
[gammab_s_b, dYb_s_b, gammaf_s_b, dYf_s_b] = resolve_gammab(gamma_s_b, dY);
[Dwxb_b, Dwyb_b, Dwzb_b] = resolve_IW(Dwx, Dwy, Dwz);
[Dwxbf_b, Dwybf_b, Dwzbf_b] = resolve_IWf(Dwx, Dwy, Dwz);
nr1b_s_b = sqrt(sum(dYb_s_b.^2,2));
norb_s_b = dYb_s_b ./ nr1b_s_b;
nr1b_sf_b = sqrt(sum(dYf_s_b.^2,2));
norb_sf_b = dYf_s_b ./ nr1b_sf_b;
[gammabnb_s_b, dYbnb_s_b] = resolve_gammanb(gamma_snb_b, dYnb);
[Dwxb_nb_b, Dwyb_nb_b, Dwzb_nb_b] = resolve_IW(Dwx_nb, Dwy_nb, Dwz_nb);
nr1nb_s_b = sqrt(sum(dYbnb_s_b.^2,2));
nornb_s_b = dYbnb_s_b ./ nr1nb_s_b;
i = 1;
%alpha = 0;
for beta = -10:2:12
[b_force2t_bb(:,:,i), Fpann_Bb(:,:,i)] = gam_components_body2(norb_s_b, gammab_s_b(:,i), (alpha) * (pi/180), beta * (pi/180),
v, dens, ...
   nr1b_s_b, -Dwxb_b, -Dwyb_b, -Dwzb_b);
[b_force2t_bbNOT(:,:,i), Fpann_BbUSE(:,:,i)] = gam_components_body2(norb_s_b, -gammab_s_b(:,i), (alpha) * (pi/180),...
   beta * (pi/180), v, dens, nr1b_s_b, -Dwxb_b, Dwyb_b, -Dwzb_b);
[nb_force2t_bb(:,:,i), Fpann_NBb(:,:,i)] = gam_components_body2(nornb_s_b, gammabnb_s_b(:,i), (alpha) * (pi/180), beta *
(pi/180), v, ...
   dens, nr1nb_s_b, -Dwxb_nb_b, -Dwyb_nb_b, -Dwzb_nb_b);
[nb_force2t_bbNOT(:,:,i), Fpann_NBbUSE(:,:,i)] = gam_components_body2(nornb_s_b, -gammabnb_s_b(:,i), (alpha) * (pi/180),
beta * (pi/180), v, ...
   dens, nr1nb_s_b, Dwxb_nb_b, Dwyb_nb_b, Dwzb_nb_b);
[bf_force2t_bb(:,:,i), Fpann_BFb(:,:,i)] = gam_components_body2(norb_sf_b, gammaf_s_b(:,i), (alpha) * (pi/180), beta *
(pi/180), v, dens, ...
   nr1b_sf_b, -Dwxbf_b, -Dwybf_b, -Dwzbf_b);
[bf_force2t_bbNOT(:,:,i), Fpann_BFbUSE(:,:,i)] = gam_components_body2(norb_sf_b, -gammaf_s_b(:,i), (alpha) * (pi/180), beta
* (pi/180), v, dens, ...
   nr1b_sf_b, Dwxbf_b, Dwybf_b, Dwzbf_b);
BodyToWind=[cos(beta*(pi/180))*cos(alpha*(pi/180)), sin(beta*(pi/180)), cos(beta*(pi/180))*sin(alpha*(pi/180));...
            cos(alpha*(pi/180))*sin(beta*(pi/180)), cos(beta*(pi/180)), -sin(beta*(pi/180))*sin(alpha*(pi/180));...
             -sin(alpha*(pi/180)),                  0,                  cos(alpha*(pi/180))];
```

A3. MATLAB function betaloop2.m

```
B_force2t_bb(:,:,i) = permute(b_force2t_bb(:,:,i), [2,1,3]);
NB_force2t_bb(:,:,i) = permute(nb_force2t_bb(:,:,i), [2,1,3]);
BF_force2t_bb(:,:,i) = permute(bf_force2t_bb(:,:,i), [2,1,3]);
BtoW_b2t_bb(i,:)= BodyToWind*(B_force2t_bb(:,:,i));
BtoW_nb2t_bb(i,:)= BodyToWind*(NB_force2t_bb(:,:,i));
BtoW_bf2t_bb(i,:)= BodyToWind*(BF_force2t_bb(:,:,i));
i = i+1;
end
Lb_bt = BtoW_b2t_bb(:,3)';
Lnb_bt = BtoW_nb2t_bb(:,3)';
Lbf_bt = BtoW_bf2t_bb(:,3)';
Db_bt = BtoW_b2t_bb(:,1)';
Dnb_bt = BtoW_nb2t_bb(:,1)';
Dbf_bt = BtoW_bf2t_bb(:,1)';
Cb_bt = (BtoW_b2t_bb(:,2)');
Cnb_bt = (BtoW_nb2t_bb(:,2)');
Cbf_bt = (BtoW_bf2t_bb(:,2)');
% moments ------------------------------------------------------------
refpt2 = [com*c_r1 - 0.0017,0,0.084]; % wing apex pnt i.e. panel plot beginning coordinate
refptbf2 = [c_r1-(com*c_r1)- 0.0017,0,0.084];
[Brteb, BFrteb] = resolve_rb(r1(:,:), r2(:,:));
[NBrte] = resolve_rnb(r1nb(:,:), r2nb(:,:));

c4b2 = Brteb - (ones(size(dYb_s(:,1))) .* refpt2);
c4nb2 = NBrte - (ones(size(dYbnb_s(:,1))) .* refpt2);
c4bf2 = BFrteb - (ones(size(dYf_s(:,1))) .* refptbf2);
i = 1;
for beta = -10:2:12
WindToBody2 = [cos(beta*(pi/180))*cos(alpha*(pi/180)), -cos(alpha*(pi/180))*sin(beta*(pi/180)), -sin(alpha*(pi/180));...
               sin(beta*(pi/180)),                     cos(beta*(pi/180)),                      0;...
           cos(beta*(pi/180))*sin(alpha*(pi/180)), -sin(beta*(pi/180))*sin(alpha*(pi/180)), -cos(alpha*(pi/180))];
        Fpann_Bb2(:,:,i) = permute(Fpann_Bb(:,:,i), [2,1,3]);
        Fpann_Bb3(:,:,i) = WindToBody2*Fpann_Bb2(:,:,i);
        Fpann_Bb4(:,:,i) = permute(Fpann_Bb3(:,:,i), [2,1,3]);

        Fpann_NBb2(:,:,i) = permute(Fpann_NBb(:,:,i), [2,1,3]);
        Fpann_NBb3(:,:,i) = WindToBody2*Fpann_NBb2(:,:,i);
        Fpann_NBb4(:,:,i) = permute(Fpann_NBb3(:,:,i), [2,1,3]);

        Fpann_BFb2(:,:,i) = permute(Fpann_BFb(:,:,i), [2,1,3]);
        Fpann_BFb3(:,:,i) = WindToBody2*Fpann_BFb2(:,:,i);
        Fpann_BFb4(:,:,i) = permute(Fpann_BFb3(:,:,i), [2,1,3]);
i = i+1;
end
if sum(Fpann_Bb4(:,1,:)) > 0
    Fpann_Bb4(:,1,:) = (Fpann_Bb4(:,1,:))*-1;
end
if sum(Fpann_NBb4(:,1,:)) > 0
    Fpann_NBb4(:,1,:) = (Fpann_Bb4(:,1,:))*-1;
end
if sum(Fpann_BFb4(:,1,:)) > 0
    Fpann_BFb4(:,1,:) = (Fpann_BFb4(:,1,:))*-1;
end
```

A3. MATLAB function betaloop2.m (continued)

```
if sum(Fpann_BFb(:,1,:)) < 0
   Fpann_BFb4(:,1,:) = (Fpann_BFb4(:,1,:))*-1;
end

for i = 1:12
M_body(:,:,i) = cross(c4b2, Fpann_Bb(:,:,i));
MbodySum(:,:,i) = cross(sum(c4b2,1), sum(Fpann_Bb(:,:,i),1));
M_nocbody(:,:,i) = cross(c4nb2, Fpann_NBb(:,:,i));
M_bodyflap(:,:,i) = cross(c4bf2, Fpann_BFb(:,:,i));

M_bodytestBb(:,:,i) = cross(c4b2,Fpann_Bb4(:,:,i));
M_bodytestNBb(:,:,i) = cross(c4nb2,Fpann_NBb4(:,:,i));
M_bodytestBFb(:,:,i) = cross(c4bf2,Fpann_BFb4(:,:,i));
end

MbodytestB = squeeze(sum(M_body,1));
MbodytestNB = squeeze(sum(M_nocbody,1));
MbodytestBF = squeeze(sum(M_bodyflap,1));
Mbody_Test = (MbodytestB + MbodytestNB)./2 + MbodytestBF;
Mbody_Test2 = ( squeeze(sum(M_bodytestBb,1)) + squeeze(sum(M_bodytestNBb,1)) )./2 + squeeze(sum(M_bodytestBFb,1));

% ------------------------------------------------------------------------
% wing
dyr = rr2(:,:) - rr1(:,:);
dYr = dyr';
nr1w = sqrt(sum(dYr.^2,2));
norw = dYr ./ nr1w;
i = 1;
for beta = -10:2:12
Alpha = alpha - 2;
rhsr_b(:,i) = boundarycondition(v, qr, (alpha - 2) * (pi/180), beta * (pi/180), rr, rr1, rr2);
gamr_s_b(:,i) = w2 \ -rhsr_b(:,i);
[w_force_b(:,:,i), Fpann_Wb(:,:,i)] = gam_components_wing2(norw, gamr_s_b(:,i), (alpha-2) * (pi/180), beta * (pi/180), ...
   v, dens, nr1w, -Dwxr, -Dwyr, -Dwzr);
BodyToWind2=[cos(beta*(pi/180))*cos((alpha - 2)*(pi/180)),sin(beta*(pi/180)),cos(beta*(pi/180))*sin((alpha - 2)*(pi/180));...
        cos((alpha - 2)*(pi/180))*sin(beta*(pi/180)),cos(beta*(pi/180)),-sin(beta*(pi/180))*sin((alpha - 2)*(pi/180));...
            -sin((alpha - 2)*(pi/180)),                  0,                  cos((alpha - 2)*(pi/180))];
W_force_b(:,:,i) = permute(w_force_b(:,:,i), [2,1,3]);
BtoW_w_b(i,:)= BodyToWind2*(W_force_b(:,:,i));
FZw_b(:,i) = w_force_b(:,3,i);
FYw_b(:,i) = w_force_b(:,2,i);
FXw_b(:,i) = w_force_b(:,1,i);
i = i+1;
end
Lwing_bt = BtoW_w_b(:,3)';
Dwing_bt = -BtoW_w_b(:,1)'; % correction factor...??
Cwing_bt = (BtoW_w_b(:,2)');

L_bt = (Lwing_bt) + (Lb_bt + Lnb_bt)/2 + Lbf_bt;
D_bt = Dwing_bt + (Db_bt + Dnb_bt)/2+ Dbf_bt;
Y_bt = (Cwing_bt + Cb_bt + Cnb_bt + Cbf_bt);
```

A3. MATLAB function betaloop2.m (continued)

```
% moments ------------------------------------------------------------------
rte = (rr1(:,:) + rr2(:,:))./2;
rTe = rte';
refptw2 = [(com*c_r1)-0.0017,-s1,0.082]; % (c_r1 - c_t1)-
refptw3 = [(com*c_r1)-0.0017,s1,0.082];
c4w2 = rTe;
for i = 1:48
c4w2(i,:) = rTe(i,:) - (ones(size(dYr(i,1))) .* refptw2);
end
for i = 57:104
c4w2(i,:) = rTe(i,:) - (ones(size(dYr(i,1))) .* refptw3);
end
for i = 49:58
c4w2(i,:) = 0;
end

i = 1;
for beta = -10:2:12
WindToBody2 = [cos(beta*(pi/180))*cos(alpha*(pi/180)), -cos(alpha*(pi/180))*sin(beta*(pi/180)), -sin(alpha*(pi/180));...
               sin(beta*(pi/180)),                     cos(beta*(pi/180)),                      0;...
             cos(beta*(pi/180))*sin(alpha*(pi/180)), -sin(beta*(pi/180))*sin(alpha*(pi/180)), -cos(alpha*(pi/180))];
        Fpann_Wb_2(:,:,i) = permute(Fpann_Wb(:,:,i), [2,1,3]);
        Fpann_Wb_3(:,:,i) = WindToBody2*Fpann_Wb_2(:,:,i);
        Fpann_Wb_4(:,:,i) = permute(Fpann_Wb_3(:,:,i), [2,1,3]);
i = i+1;
end
if sum(Fpann_Wb_4(:,1,:)) > 0
    Fpann_Wb_4(:,1,:) = (Fpann_Wb_4(:,1,:))*-1;
end
squeeze(sum(Fpann_Wb_4))

for i = 1:12
M_wing3(:,:,i) = cross(c4w2, Fpann_Wb_4(:,:,i));
end
Mwing_Test2 = squeeze(sum(M_wing3));
```

A3. MATLAB function betaloop2.m (concluded)

```
%calculating freestream normal at collocation point
function BC = boundarycondition(v, q, alpha, beta, r ,r1 ,r2)
%calculating normals
 [aa bb cc] = size(q);
n = cross(r(:,:) - r1(:,:), r(:,:) - r2(:,:));
n1 = sqrt(sum(n.^2));
no = n ./ n1;
V = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
bc1 = sum( no' .* v_inf,2); %steady state boundary condition
BC = -bc1;
end
```

A4. MATLAB function boundarycondition.m

```
% adapted from Melin
function [forceb, Fpan] = gam_components_body2(no, gamma, alpha, beta, v, dens, nr, Dwx, Dwy, Dwz)
G1 = no(:,1).*gamma;
G2 = no(:,2).*gamma;
G3 = no(:,3).*gamma;
G(:,1) = G1;
G(:,2) = G2;
G(:,3) = G3;
IW(:,1)=Dwx*gamma;
IW(:,2)=Dwy*gamma;
IW(:,3)=Dwz*gamma;
wind = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)] .* ones(size(gamma));
wind = wind - IW;
Fppan = dens * cross(wind,G);
% force per panel
Fpan(:,1) = Fppan(:,1).* nr;
Fpan(:,2) = Fppan(:,2).* nr;
Fpan(:,3) = Fppan(:,3).* nr;
forceb = sum(Fpan,1);
end
```

A5. MATLAB function gam_components_body2.m

```
% gamma components function
function [force, Fpan2] = gam_components_wing2(nor, gam, alpha, beta, v, dens, nr1, Dwxr, Dwyr, Dwzr)
G1r = nor(:,1).*gam;
G2r = nor(:,2).*gam;
G3r = nor(:,3).*gam;
Gr(:,1) = G1r;
Gr(:,2) = G2r;
Gr(:,3) = G3r;
IW(:,1)=Dwxr*gam;
IW(:,2)=Dwyr*gam;
IW(:,3)=Dwzr*gam;
windr = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)] .* ones(size(gam));
windr = windr - IW;
Fppan = dens .* cross(windr,Gr);
 % force per panel
Fpan(:,1) = Fppan(:,1).* nr1;
Fpan(:,2) = Fppan(:,2).* nr1;
Fpan(:,3) = Fppan(:,3).* nr1;
for i = 49:56
   Fpan(i,:) = 0;
end
Fpan2 = Fpan;
force = sum(Fpan2,1);
end
```

A6. MATLAB function gam_components_wing2.m

```
% adapted from Melin, calculating derivatives from perturbation delta
function BC2 = boundarycondition2(v, q, alpha, beta, r ,r1 ,r2, P, Q, R, com)
%calculating normals
delta = 0.0001; % from Melin
rte = (r1(:,:) + r2(:,:))./2;
rTe = rte';
[aa bb cc] = size(q);
n = cross(r(:,:) - r1(:,:), r(:,:) - r2(:,:));
n1 = sqrt(sum(n.^2));
no = n ./ n1;
V = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
% adding in roll, pitch, and yaw rates and rotation
for i = 1:bb
Rot(i,:) = cross((rTe(bb,:) - com),[P Q R]);
end
V_inf = v_inf + Rot;
bc1 = sum( no' .* V_inf,2); %steady state boundary condition
BC2(:,1) = -bc1;

% alpha derivative column
alpha = alpha + delta;
V = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
for i = 1:bb
Rot(i,:) = cross((rTe(bb,:) - com),[P Q R]);
end
V_inf = v_inf + Rot;
bc1 = sum( no' .* V_inf,2);
BC2(:,2) = -bc1;
alpha = alpha - delta;

% beta derivative column
beta = beta + delta;
V = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
for i = 1:bb
Rot(i,:) = cross((rTe(bb,:) - com),[P Q R]);
end
V_inf = v_inf + Rot;
bc1 = sum( no' .* V_inf,2);
BC2(:,3) = -bc1;
beta = beta - delta;

% roll rate, p, derivative column
P = P + delta;
V = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
for i = 1:bb
Rot(i,:) = cross((rTe(bb,:) - com),[P Q R]);
end
V_inf = v_inf + Rot;
bc1 = sum( no' .* V_inf,2);
BC2(:,4) = -bc1;
P = P - delta;
```

A7. MATLAB function boundarycondition2.m

```
% pitch rate, q, derivative column
Q = Q + delta;
V = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
for i = 1:bb
Rot(i,:) = cross((rTe(bb,:) - com),[P Q R]);
end
V_inf = v_inf + Rot;
bc1 = sum( no' .* V_inf,2);
BC2(:,5) = -bc1;
Q = Q - delta;

% yaw rate, r, derivative column
R = R + delta;
V = v .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
for i = 1:bb
Rot(i,:) = cross((rTe(bb,:) - com),[P Q R]);
end
V_inf = v_inf + Rot;
bc1 = sum( no' .* V_inf,2);
BC2(:,6) = -bc1;
R = R - delta;

% velocity, u, derivative column
%v = v + delta;
V = (v+delta) .* [cos(alpha) * cos(beta) -cos(alpha) * sin(beta) sin(alpha)];
v_inf = V .* ones(bb,1);
for i = 1:bb
Rot(i,:) = cross((rTe(bb,:) - com),[P Q R]);
end
V_inf = v_inf + Rot;
bc1 = sum( no' .* V_inf,2);
BC2(:,7) = -bc1;
%v = v - delta;
end
```

A7. MATLAB function boundarycondition2.m (concluded)

```
% glider for Simulink
% aerolib for blocks
% constants ----------------------------------------------------------------
deg2rad = pi/180;

% configuration -------------------------------------------------------------
S_ref = 0.2802;          % Reference area [m^2]
d_ref = 1.0367;          % Reference length [m]
b_ref = 0.4718;          % Reference Span [m]
massBody = 6;             % Mass of Body [Kg]
mass = massBody;
len_veh = 0.9589;            % Vehicle Length
x_ref = 0.5639;          % Reference point from nose
x_cg = 0.5517;          % Center of Gravity (Full)
Ixx = 1/8*mass*(S_ref/pi);
Iyy = 1/3*mass*(d_ref/2)^2;
Izz = Iyy;
Inertia = diag([Ixx Iyy Izz]);

% initial conditions --------------------------------------------------------
alpha_0 = 0.0268*-1; % glide angle, rad
alpha0 = alpha_0;
Vmw =  15*[cos(alpha_0);0;sin(alpha_0)]; % Velocity in Body Axes
pm_0 = [0;0;0]; % Initial angular rates (rad/sec)
xme_0 = [0; 0; -127]; % Initial Position
Euler_0 = [0; -0.3137+alpha_0; 0];
LongLat0 = [-122.3896; 37.6272]; % from hl20 example
Xme_ref = 0;

% aerodynamic coefficients --------------------------------------------------
% basic >>
alpha_vecs = al;
beta_vecs = be;
[bev,alv] = meshgrid(beta_vecs, alpha_vecs);
Be = bev(:); Al = alv(:);
% calculating look up tables >> CX = -CA, CZ = -CN, CY = CY
PolyBasicForces = [CX_coeff_New', CZ_coeff_New'];
temp = [ones(length(Al),1) Be Al Be.^2 Be.*Al Al.^2 Be.^3 Be.^2.*Al Be.*Al.^2 Al.^3 Be.^4 Be.^3.*Al Be.^2.*Al.^2 ...
   Be.*Al.^3 Al.^4] * PolyBasicForces;
CX_0 = reshape(temp(:,1),length(alpha_vecs),length(beta_vecs));
CZ_0 = reshape(temp(:,2),length(alpha_vecs),length(beta_vecs));

PolyBasicMoments = [Cm_coeff_New', Cn_coeff_New'];
temp = [ones(length(Al),1) Be Al Be.^2 Be.*Al Al.^2 Be.^3 Be.^2.*Al Be.*Al.^2 Al.^3 Be.^4 Be.^3.*Al Be.^2.*Al.^2 ...
   Be.*Al.^3 Al.^4] * PolyBasicMoments;
Cm_0 = reshape(temp(:,1),length(alpha_vecs),length(beta_vecs));
Cn_0 = reshape(temp(:,2),length(alpha_vecs),length(beta_vecs));

% Side force and roll stability derivatives
CY_Beta = -0.0103/2; % per degree
Cl_Beta = -0.0031;
```

A8. MATLAB script SimlnkGlider.m

```
% Positive Body Flap >>
polyCoeffBFp = [CX_delBFp_coeff2', Cm_delBFp_coeff2', CZ_delBFp_coeff2'];
temp = [al'.^4 al'.^3 al'.^2 al'.^1 ones(length(alpha_vecs),1)] * polyCoeffBFp;
CX_dbfp = temp(:,1);
Cm_dbfp = temp(:,2);
CZ_dbfp = temp(:,3);


% Symmetric Wing Flaps >> elevator
polyCoeffE = [CX_delE_coeff2', Cm_delE_coeff2', CZ_delE_coeff2'];
temp = [al'.^4 al'.^3 al'.^2 al'.^1 ones(length(alpha_vecs),1)] * polyCoeffE;
CX_de = temp(:,1);
Cm_de = temp(:,2);
CZ_de = temp(:,3);

% Differential Wing Flaps >> aileron
polyCoeffA = [CX_delA_coeff2', Cm_delA_coeff2', CZ_delA_coeff2', Cn_delA_coeff2', CY_delA_coeff2', Cl_delA_coeff2'];
temp = [al'.^4 al'.^3 al'.^2 al'.^1 ones(length(alpha_vecs),1)] * polyCoeffA;
CX_da = temp(:,1);
Cm_da = temp(:,2);
CZ_da = temp(:,3);
Cn_da = temp(:,6);
CY_da = temp(:,5);
Cl_da = temp(:,4);

% adding in the beta contributions >>
polyCoeffAb = [CX_delAb_coeff', Cm_delAb_coeff', CZ_delAb_coeff', Cn_delAb_coeff', CY_delAb_coeff', Cl_delAb_coeff'];
temp = [ones(length(Al),1) Be Al Be.^2 Be.*Al Al.^2 Be.^3 Be.^2.*Al Be.*Al.^2 Al.^3 Be.^4 Be.^3.*Al Be.^2.*Al.^2 ...
   Be.*Al.^3 Al.^4] * polyCoeffAb;
CX_dab = reshape(temp(:,1),length(alpha_vecs),length(beta_vecs));
Cm_dab = reshape(temp(:,2),length(alpha_vecs),length(beta_vecs));
CZ_dab = reshape(temp(:,3),length(alpha_vecs),length(beta_vecs));
Cn_dab = reshape(temp(:,4),length(alpha_vecs),length(beta_vecs));
CY_dab = reshape(temp(:,5),length(alpha_vecs),length(beta_vecs));
Cl_dab = reshape(temp(:,6),length(alpha_vecs),length(beta_vecs));

% Damping Coefficients (per rad/sec) >>
alpha_vec_damp = -2:2:20;
Cm_q = Cmq;
Cl_p = Clp;
Cn_p = Cnp;
Cl_r = Clr;
Cn_r = Cnr;
```

A8. MATLAB script SimlnkGlider.m (concluded)

APPENDIX B – DATA TABLES

Table B1. C_X for basic configuration

	$C_{X,0}$ for β°					
α°	-10	-8	-6	-4	-2	0
-2	-0.0775	-0.0774	-0.0773	-0.0772	-0.0771	-0.0771
0	-0.0758	-0.0750	-0.0744	-0.0739	-0.0736	-0.0736
2	-0.0731	-0.0719	-0.0709	-0.0702	-0.0698	-0.0696
4	-0.0726	-0.0712	-0.0700	-0.0692	-0.0687	-0.0685
6	-0.0656	-0.0643	-0.0632	-0.0624	-0.0619	-0.0617
8	-0.0578	-0.0567	-0.0559	-0.0552	-0.0548	-0.0547
10	-0.0484	-0.0478	-0.0474	-0.0470	-0.0468	-0.0468
12	-0.0370	-0.0372	-0.0373	-0.0374	-0.0374	-0.0375
14	-0.0232	-0.0244	-0.0252	-0.0258	-0.0262	-0.0263
16	-0.0069	-0.0092	-0.0109	-0.0122	-0.0129	-0.0132
18	0.0092	0.0056	0.0028	0.0009	-0.0002	-0.0006
20	0.0224	0.0174	0.0136	0.0108	0.0092	0.0086

	$C_{X,0}$ for β°					
α°	2	4	6	8	10	12
-2	-0.0771	-0.0772	-0.0773	-0.0774	-0.0775	-0.0776
0	-0.0737	-0.0739	-0.0744	-0.0750	-0.0758	-0.0767
2	-0.0698	-0.0702	-0.0709	-0.0719	-0.0731	-0.0745
4	-0.0687	-0.0692	-0.0700	-0.0712	-0.0726	-0.0742
6	-0.0619	-0.0624	-0.0632	-0.0643	-0.0656	-0.0672
8	-0.0548	-0.0552	-0.0559	-0.0567	-0.0578	-0.0590
10	-0.0468	-0.0470	-0.0474	-0.0479	-0.0484	-0.0490
12	-0.0374	-0.0374	-0.0373	-0.0372	-0.0370	-0.0367
14	-0.0262	-0.0259	-0.0253	-0.0244	-0.0232	-0.0218
16	-0.0130	-0.0122	-0.0110	-0.0092	-0.0069	-0.0041
18	-0.0002	0.0009	0.0028	0.0056	0.0091	0.0135
20	0.0092	0.0108	0.0136	0.0174	0.0223	0.0283

Table B2. C_Z for basic configuration

	$C_{Z,0}$ for β°					
α°	-10	-8	-6	-4	-2	0
-2	0.2124	0.2143	0.2158	0.2169	0.2175	0.2178
0	0.1045	0.1059	0.1070	0.1079	0.1083	0.1085
2	-0.0048	-0.0038	-0.0031	-0.0026	-0.0023	-0.0022
4	-0.1158	-0.1154	-0.1151	-0.1149	-0.1147	-0.1147
6	-0.2260	-0.2262	-0.2263	-0.2264	-0.2264	-0.2264
8	-0.3366	-0.3373	-0.3379	-0.3382	-0.3385	-0.3386
10	-0.4612	-0.4626	-0.4636	-0.4643	-0.4647	-0.4649
12	-0.5737	-0.5756	-0.5770	-0.5780	-0.5787	-0.5789
14	-0.6686	-0.6709	-0.6727	-0.6739	-0.6747	-0.6749
16	-0.7597	-0.7623	-0.7644	-0.7658	-0.7667	-0.7670
18	-0.8401	-0.8429	-0.8452	-0.8467	-0.8477	-0.8480
20	-0.9016	-0.9045	-0.9068	-0.9085	-0.9094	-0.9098

	$C_{Z,0}$ for β°					
α°	2	4	6	8	10	12
-2	0.2175	0.2169	0.2158	0.2143	0.2124	0.2101
0	0.1083	0.1079	0.1070	0.1059	0.1045	0.1027
2	-0.0023	-0.0026	-0.0031	-0.0038	-0.0048	-0.0059
4	-0.1147	-0.1149	-0.1151	-0.1154	-0.1158	-0.1163
6	-0.2264	-0.2264	-0.2263	-0.2262	-0.2260	-0.2258
8	-0.3385	-0.3382	-0.3379	-0.3373	-0.3366	-0.3357
10	-0.4647	-0.4643	-0.4636	-0.4626	-0.4612	-0.4596
12	-0.5787	-0.5780	-0.5770	-0.5756	-0.5737	-0.5714
14	-0.6747	-0.6739	-0.6727	-0.6709	-0.6686	-0.6659
16	-0.7667	-0.7658	-0.7644	-0.7623	-0.7597	-0.7565
18	-0.8477	-0.8467	-0.8452	-0.8429	-0.8401	-0.8366
20	-0.9094	-0.9085	-0.9068	-0.9045	-0.9016	-0.8980

Table B3. C_m for basic configuration

	$C_{m,0}$ for β°					
α°	-10	-8	-6	-4	-2	0
-2	0.0696	0.0716	0.0731	0.0739	0.0744	0.0744
0	0.0380	0.0399	0.0414	0.0422	0.0428	0.0428
2	0.0075	0.0094	0.0109	0.0117	0.0123	0.0125
4	-0.0220	-0.0201	-0.0187	-0.0178	-0.0172	-0.0169
6	-0.0508	-0.0491	-0.0476	-0.0468	-0.0460	-0.0457
8	-0.0793	-0.0775	-0.0761	-0.0752	-0.0744	-0.0740
10	-0.1074	-0.1057	-0.1043	-0.1034	-0.1025	-0.1020
12	-0.1356	-0.1339	-0.1324	-0.1315	-0.1305	-0.1299
14	-0.1639	-0.1623	-0.1608	-0.1598	-0.1588	-0.1581
16	-0.1927	-0.1910	-0.1895	-0.1885	-0.1874	-0.1866
18	-0.2220	-0.2204	-0.2189	-0.2178	-0.2166	-0.2157
20	-0.2521	-0.2505	-0.2490	-0.2478	-0.2466	-0.2455

	$C_{m,0}$ for β°					
α°	2	4	6	8	10	12
-2	0.0738	0.0727	0.0711	0.0689	0.0663	0.0631
0	0.0424	0.0415	0.0400	0.0382	0.0358	0.0330
2	0.0121	0.0114	0.0102	0.0085	0.0065	0.0039
4	-0.0171	-0.0177	-0.0187	-0.0201	-0.0219	-0.0241
6	-0.0458	-0.0462	-0.0470	-0.0481	-0.0497	-0.0516
8	-0.0739	-0.0742	-0.0748	-0.0757	-0.0770	-0.0786
10	-0.1018	-0.1019	-0.1023	-0.1030	-0.1040	-0.1053
12	-0.1296	-0.1296	-0.1298	-0.1302	-0.1310	-0.1321
14	-0.1576	-0.1574	-0.1574	-0.1577	-0.1581	-0.1589
16	-0.1860	-0.1856	-0.1854	-0.1854	-0.1857	-0.1862
18	-0.2149	-0.2144	-0.2140	-0.2138	-0.2137	-0.2140
20	-0.2446	-0.2439	-0.2433	-0.2429	-0.2426	-0.2425

Table B4. C_n for basic configuration

	$C_{n,0}$ for β°					
α°	-10	-8	-6	-4	-2	0
-2	-0.0155	-0.0117	-0.0079	-0.0041	-0.0003	0.0034
0	-0.0133	-0.0103	-0.0073	-0.0044	-0.0014	0.0014
2	-0.0120	-0.0095	-0.0071	-0.0047	-0.0023	6.720e-06
4	-0.0119	-0.0096	-0.0074	-0.0053	-0.0031	-0.0009
6	-0.0130	-0.0106	-0.0083	-0.0060	-0.0037	-0.0015
8	-0.0149	-0.0122	-0.0095	-0.0068	-0.0041	-0.0015
10	-0.0178	-0.0144	-0.0110	-0.0076	-0.0043	-0.0010
12	-0.0216	-0.0172	-0.0128	-0.0085	-0.0043	-0.0001
14	-0.0262	-0.0205	-0.0150	-0.0095	-0.0040	0.0012
16	-0.0315	-0.0244	-0.0174	-0.0105	-0.0036	0.0032
18	-0.0376	-0.0288	-0.0201	-0.0114	-0.0029	0.0056
20	-0.0443	-0.0336	-0.0230	-0.0124	-0.0019	0.0084

	$C_{n,0}$ for β°					
α°	2	4	6	8	10	12
-2	0.0072	0.0110	0.0149	0.0188	0.0227	0.0268
0	0.0044	0.0074	0.0104	0.0135	0.0166	0.0198
2	0.0023	0.0047	0.0072	0.0096	0.0122	0.0148
4	0.0012	0.0033	0.0055	0.0077	0.0100	0.0123
6	0.0007	0.0029	0.0052	0.0075	0.0098	0.0122
8	0.0010	0.0036	0.0063	0.0089	0.0116	0.0143
10	0.0021	0.0054	0.0086	0.0119	0.0152	0.0185
12	0.0040	0.0082	0.0123	0.0165	0.0207	0.0249
14	0.0066	0.0120	0.0173	0.0226	0.0279	0.0333
16	0.0100	0.0167	0.0235	0.0302	0.0369	0.0437
18	0.0140	0.0224	0.0308	0.0392	0.0476	0.0559
20	0.0188	0.0291	0.0393	0.0495	0.0597	0.0699

Table B5. Incremental force and moment coefficients per degree of deflection of body flap

α°	$C_{X_{\delta_{bf}}}$	$C_{m_{\delta_{bf}}}$	$C_{Z_{\delta_{bf}}}$
-2	0.0220	-0.4011	-0.2425
0	0.0208	-0.4056	-0.2452
2	0.0197	-0.4091	-0.2472
4	0.0190	-0.4116	-0.2487
6	0.0185	-0.4131	-0.2496
8	0.0183	-0.4136	-0.2499
10	0.0185	-0.4131	-0.2496
12	0.0189	-0.4116	-0.2487
14	0.0196	-0.4091	-0.2472
16	0.0206	-0.4056	-0.2452
18	0.0219	-0.4012	-0.2425
20	0.0234	-0.3958	-0.2393

Table B6. Incremental force and moment coefficients per degree of deflection of elevon

α°	$C_{X_{\delta_e}}$	$C_{m_{\delta_e}}$	$C_{Z_{\delta_e}}$
-2	0.0152	-0.2117	-0.1872
0	0.0139	-0.2140	-0.1889
2	0.0129	-0.2158	-0.1903
4	0.0121	-0.2171	-0.1912
6	0.0116	-0.2179	-0.1916
8	0.0115	-0.2182	-0.1915
10	0.0116	-0.2179	-0.1910
12	0.0121	-0.2171	-0.1901
14	0.0129	-0.2158	-0.1886
16	0.0139	-0.2140	-0.1868
18	0.0152	-0.2117	-0.1844
20	0.0168	-0.2090	-0.1817

Table B7. Incremental force and moment coefficients per degree of deflection of aileron

α°	$C_{X_{\delta_a}}$	$C_{m_{\delta_a}}$	$C_{Z_{\delta_a}}$	$C_{n_{\delta_a}}$	$C_{Y_{\delta_a}}$	$C_{l_{\delta_a}}$
-2	-0.0063	-0.0959	-0.1108	-0.0375	0.0352	0.0446
0	-0.0049	-0.0965	-0.1108	-0.0360	0.0355	0.0441
2	-0.0032	-0.0965	-0.1108	-0.0349	0.0358	0.0435
4	-0.0011	-0.0965	-0.1105	-0.0338	0.0358	0.0426
6	0.0014	-0.0962	-0.1100	-0.0323	0.0355	0.0421
8	0.0037	-0.0956	-0.1091	-0.0312	0.0349	0.0409
10	0.0066	-0.0951	-0.1080	-0.0297	0.0340	0.0401
12	0.0095	-0.0939	-0.1065	-0.0286	0.0329	0.0389
14	0.0126	-0.0928	-0.1048	-0.0275	0.0317	0.0378
16	0.0155	-0.0913	-0.1028	-0.0263	0.0303	0.0366
18	0.0183	-0.0896	-0.1008	-0.0252	0.0286	0.0352
20	0.0212	-0.0879	-0.0985	-0.0243	0.0266	0.0340

APPENDIX C – SUPPLEMENTARY INFORMATION

```
Operating point search report:
---------------------------------

 Operating point search report for the Model GliderFlightAnalysis.
 (Time-Varying Components Evaluated at time t=0)

Operating point specifications were successfully met.
States:
----------
(1.) phi
      x:      -0.00064      dx:     -2.16e-22 (0)
(2.) theta
      x:      -0.00134      dx:      4.12e-22 (0)
(3.) psi
      x:         0.557      dx:     -4.08e-24 (0)
(4.) p
      x:     -2.16e-22      dx:      -1.7e-14 (0)
(5.) q
      x:      4.12e-22      dx:      1.16e-11 (0)
(6.) r
      x:     -3.82e-24      dx:        -2e-15 (0)
(7.) Ubody
      x:          6.62      dx:      1.93e-14 (0)
(8.) Vbody
      x:         -4.12      dx:      -1.4e-14 (0)
(9.) Wbody
      x:       -0.0115      dx:     -2.76e-13 (0)
(10.) Xe
      x:     -3.05e-14      dx:           7.8
(11.) Ye
      x:     -1.57e-13      dx:     -4.58e-13 (0)
(12.) Ze
      x:          -127      dx:      4.29e-14 (0)
(13.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on angular rates/Hpgw/pgw_p
      x:         0.797      dx:        -0.414
      x:             0      dx:             0
(14.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on angular rates/Hqgw/qgw_p
      x:          1.57      dx:         -1.09
      x:             0      dx:             0
(15.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on angular rates/Hrgw/rgw_p
      x:          1.18      dx:          -1.1
      x:             0      dx:             0
(16.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hugw(s)/ug_p
      x:          1.91      dx:       -0.0453
      x:             0      dx:             0
(17.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hvgw(s)/vg_p1
      x:             0      dx:             0
      x:             0      dx:             0
```

Figure C1. Operating Point Search Report

```
(18.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hvgw(s)/vgw_p2
      x:          -1.4      dx:        0.0333
      x:             0      dx:             0
(19.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hwgw(s)/wg_p1
      x:      2.01e-15      dx:     -1.05e-16
      x:     -4.98e-16      dx:      6.18e-18
(20.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hwgw(s)/wg_p2
      x:         -2.27      dx:         0.118
      x:             0      dx:      4.52e-18

Inputs:
----------
(1.) GliderFlightAnalysis/AileronCmd
      u:         0.334    [-0.349 0.349]
(2.) GliderFlightAnalysis/ElevatorCmd
      u:        0.0841    [-0.349 0.349]
(3.) GliderFlightAnalysis/BodyFlapCmd
      u:        -0.289    [-0.314 0.384]

Outputs:
----------
(1.) GliderFlightAnalysis/StatesOut
      y:     -3.05e-14    [-Inf Inf]
      y:     -1.57e-13    [-Inf Inf]
      y:          -127    [-Inf Inf]
      y:      -0.00064    [-Inf Inf]
      y:      -0.00134    [-Inf Inf]
      y:         0.557    [-Inf Inf]
      y:          6.62    [-Inf Inf]
      y:         -4.12    [-Inf Inf]
      y:       -0.0115    [-Inf Inf]
      y:     -2.16e-22    [-Inf Inf]
      y:      4.12e-22    [-Inf Inf]
      y:     -3.82e-24    [-Inf Inf]
      y:      -1.7e-14    [-Inf Inf]
      y:      1.16e-11    [-Inf Inf]
      y:        -2e-15    [-Inf Inf]
      y:      1.93e-14    [-Inf Inf]
      y:      -1.4e-14    [-Inf Inf]
      y:     -2.76e-13    [-Inf Inf]

 Operating point for the Model GliderFlightAnalysis.
 (Time-Varying Components Evaluated at time t=0)

States:
----------
(1.) phi
      x: -0.00064
(2.) theta
      x: -0.00134
(3.) psi
      x: 0.557
(4.) p
      x: -2.16e-22
```

Figure C1. Operating Point Search Report (continued)

```
(5.) q
      x: 4.12e-22
(6.) r
      x: -3.82e-24
 (7.) Ubody
      x: 6.62
(8.) Vbody
      x: -4.12
(9.) Wbody
      x: -0.0115
(10.) Xe
      x: -3.05e-14
(11.) Ye
      x: -1.57e-13
(12.) Ze
      x: -127
(13.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on angular rates/Hpgw/pgw_p
      x: 0.797
      x: 0
(14.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on angular rates/Hqgw/qgw_p
      x: 1.57
      x: 0
(15.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on angular rates/Hrgw/rgw_p
      x: 1.18
      x: 0
(16.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hugw(s)/ug_p
      x: 1.91
      x: 0
(17.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hvgw(s)/vg_p1
      x: 0
      x: 0
(18.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hvgw(s)/vgw_p2
      x: -1.4
      x: 0
(19.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hwgw(s)/wg_p1
      x: 2.01e-15
      x: -4.98e-16
(20.) GliderFlightAnalysis/Environment Model/Dryden Wind Turbulence Model  (Continuous
(+q +r))/Filters on velocities/Hwgw(s)/wg_p2
      x: -2.27
      x: 0

Inputs:
----------
(1.) GliderFlightAnalysis/AileronCmd
      u: 0.334
(2.) GliderFlightAnalysis/ElevatorCmd
      u: 0.0841
(3.) GliderFlightAnalysis/BodyFlapCmd
      u: -0.289
```

Figure C1. Operating Point Search Report (concluded)